HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

BACHELORARBEIT

Herr
Eric Kurzhals

Business Intelligence Lösung für ein kommunales Entsorgungsunternehmen - Auswahl und prototypische Umsetzung -

2012

Fakultät: **Mathematik/**
Naturwissenschaften/Informatik

BACHELORARBEIT

Business Intelligence Lösung für ein kommunales Entsorgungsunternehmen - Auswahl und prototypische Umsetzung -

Autor:
Eric Kurzhals

Studiengang:
Informatik

Seminargruppe:
IF09w1-B

Erstprüfer:
Prof. Dr.-Ing. Andreas Ittner
Hochschule Mittweida (FH)

Zweitprüfer:
Dipl.-Ing. Olaf Böhm
Stadtentwässerung Dresden GmbH

Bibliografische Angaben

Kurzhals, Eric: Business Intelligence Lösung für ein kommunales Entsorgungsunternehmen – Auswahl und Umsetzung –; 54 Seiten, 38 Abbildungen, Hochschule Mittweida (FH), Fakultät Mathematik/Naturwissenschaften/Informatik

Bachelorarbeit, 2012

Referat

In der vorliegenden Bachelorarbeit wird ein prototypischer Report mit den Microsoft Analysis and Reporting Services entworfen und implementiert. Es wird dabei auf die Werkzeuge SSIS, BIDS und Analysis Services eingegangen.

Inhaltsverzeichnis

Abkürzungsverzeichnis

BI	Business Intelligence
BIDS	Business Intelligence Development Studio
BIRT	Business Intelligence and Reporting Tools
CRM	Customer Relationship Management
DBMS	Database Management System
DDS	Data Storage Server
DMX	Data Mining Extensions
DWH	Data Warehouse
EIS	Enterprise Information System
ERP	Enterprise Resource Planning
ETL	Extract Transform Load
GUI	Graphical User Interface
HOLAP	Hybrid Online Analytical Processing
LYTP	Last Year Time Period
MDX	Multidimensional Expressions
MIS	Management Information System
MOLAP	Multidimensional Online Analytical Processing
MS	Microsoft
ODS	Operational Data Store
OLAP	Online Analytical Processing
ROLAP	Relational Online Analytical Processing

SSIS	SQL Server Integration Services
SSRS	SQL Server Reporting Services
VS	Visual Studio
XBRL	Extensible Business Reporting Language
XML	Extensible Markup Language
XMLA	XML for Analysis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Die Hauptaufgabe einer Business Intelligence Lösung besteht aus der Bereitstellung der benötigten Informationen für Entscheidungsträger. Die Informationen stammen aus den unterschiedlichsten Quellen, wie zum Beispiel dem ERP, CRM oder auch aus Internetquellen.

Die Informationen eines Berichts können direkt aus den oben genannten Systemen stammen oder werden vorab in einer geeigneten Form aufbereitet gespeichert.

Business Intelligence Lösungen existieren bereits seit den 60er Jahren, wenn auch anfangs unter anderen Namen. Frühere Begriffe waren Management Informationssysteme (60er Jahre), Decision Support Systems (70er Jahre) und Executive Information Systems (80er Jahre). Erst 1989 wurde der Begriff „Business Intelligence“ von Howard Dresner eingeführt.

1.1 Problemstellung

Die Stadtentwässerung Dresden GmbH verwendet bereits die Reporting Services von Microsoft. Vorhandene Berichte werden direkt aus produktiven Daten erstellt. Dieses Vorgehen beinhaltet einige Nachteile. So benötigen einige Berichte weit über 30 Minuten der Erstellung, die Last liegt vollständig auf den produktiven Datenbankservern und die Komplexität der Abfragen ist sehr hoch.

1.2 Zielsetzung

Diese Arbeit soll eine Übersicht über den grundsätzlichen Aufbau von Business Intelligence Lösungen für Reports geben. Weiter soll eine kurze Übersicht von verschiedenen Möglichkeiten der Umsetzung, mit oder ohne eigene Datenhaltung, dargestellt werden.

Ein Prototyp wird mit einer BI-Suite erstellt. Das Beispiel wurde so gewählt, dass es leicht nachvollzogen werden kann und dennoch Besonderheiten aufweist. Es soll dem Leser in bestimmte Werkzeuge der BI-Suite einführen und einen Überblick über den Funktionsumfang geben.

1.3 Methodisches Vorgehen

Die vorliegende Arbeit besteht aus einem theoretischen und einem praktischen Teil. Der theoretische Teil wird die zum besseren Verständnis der Arbeit benötigten Grundlagen behandeln. Weiterhin werden BI-Lösungen verschiedener Hersteller und verschiedener Anforderungen verglichen.

Im praktischen Teil wird ein Prototyp eines Reports mit den Microsoft Analysis and Reporting Services erstellt. Hierbei wird besonders auf die Schwerpunkte Data Warehouse, ETL Prozess und OLAP-Entwurf eingegangen.

Abschließend werden die drei Möglichkeiten ohne Datenhaltung, Data Warhouse und OLAP Datenhaltung anhand eines Abfragedauertests miteinander verglichen und die gewonnenen Erkenntnisse zusammengefasst.

2 Grundlagen

Business Intelligence wird ein immer wichtigerer Bestandteil der betrieblichen Informationssysteme. Es kommt bei etwa 25 % der mittelständischen und sogar bei mehr als 50 % der Großunternehmen zum Einsatz[1]. Im überwiegenden Teil wird das Business Intelligence zur automatischen Erstellung von Berichten genutzt.

Die Ansätze zur Umsetzung von BI-Lösungen reichen von einfachen Reporting Systemen, die ihre Berichte direkt aus den operativen Daten generieren, bis hin zu komplexen Systemen, die ihre Datenhaltung über ein Data Warehouse organisieren.

2.1 Begriffsdefinition

Der Begriff Business Intelligence, auf deutsch etwa Geschäftsinformation, wurde Mitte der 90er Jahre von Howard Dresner, ein Analyst der Gartner Group, geprägt. In den folgenden Jahren wurden unterschiedliche Definitionen für den Begriff Business Intelligence in der Literatur verwendet, sie können auf drei Kategorien reduziert werden.[2]

2.1.1 Enges Business Intelligence

Im engeren Sinne werden Business Intelligence Systeme bezeichnet, die eine direkte Entscheidungshilfe unterstützen. Beispiele hierfür sind OLAP und MIS/EIS Systeme. Software, die der engen BI zugeordnet werden, verfügen über keine analytischen oder Datenhaltungsmodule.

2.1.2 Analyseorientiertes Business Intelligence

Unter analyseorientiertem Business Intelligence versteht man Anwendungen, bei denen der Anwender direkt mit dem System arbeitet. Er hat Zugriff auf eine Benutzeroberfläche mit interaktiven Funktionen und kann mit ihnen die Ausgabe formatieren und eingrenzen. Vertreter dieser Kategorie sind unter anderem Online Analytical Processing (OLAP) und Verfahren zum Data Mining.

---

[1] vgl. Chamoni, Gluchowski (2006) S. 52

[2] vgl. Kemper, Baars, Mehanna (2010) S. 3ff

2.1.3 Weites Business Intelligence

Im weiten Sinne werden alle Systeme unter dem Begriff Business Intelligence zusammengefasst, die für eine direkte und indirekte Entscheidungshilfe eingesetzt werden können. Die BI-Systeme i. w. S. beinhalten zusätzlich zur Präsentationsschicht eine Datenhaltung und Aufbereitung.

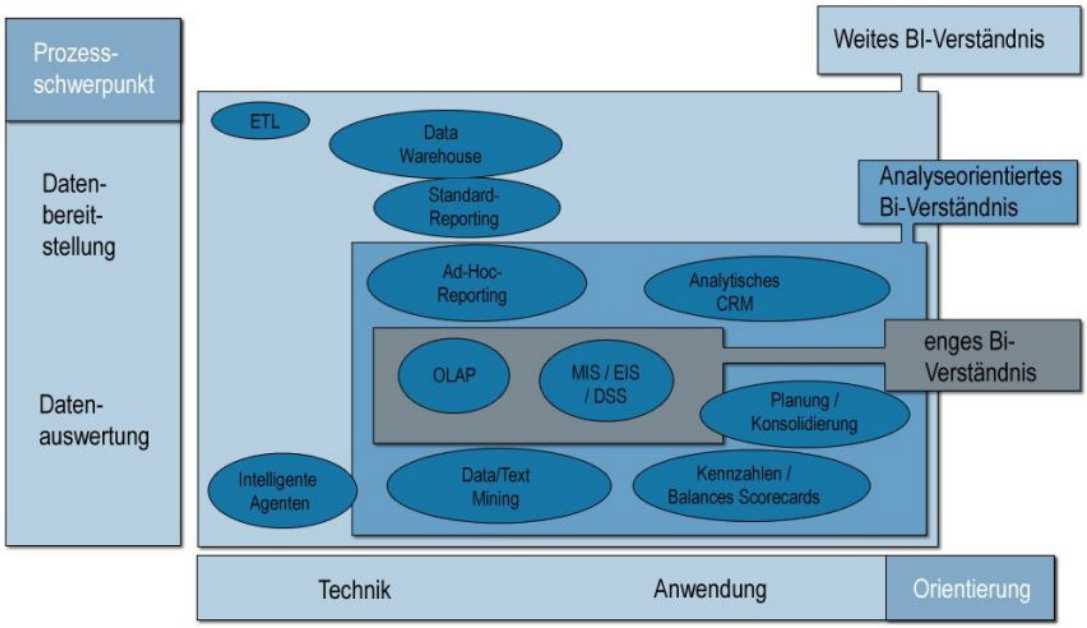

Abbildung 1 Überblick weites, enges und analyseorientiertes BI[3]

2.2 Data Warehouse

Ein Data Warehouse ist eine von den operativen Datenbanken abgekapselte Datenhaltung. Als Datenquelle können im optimalen Fall alle zur Analyse notwendigen Informationen aus den im Unternehmen eingesetzten DV-Systemen agieren. Ebenso können externe Informationsquellen (z.B. Online Dienste, Marktforschungsinstitute) eingebunden werden.

[3] Bildquelle: tecchannel.de
http://www.tecchannel.de/bild-zoom/1738998/7/362921/il-78353588462682581/ aufgerufen am 27.07.2012

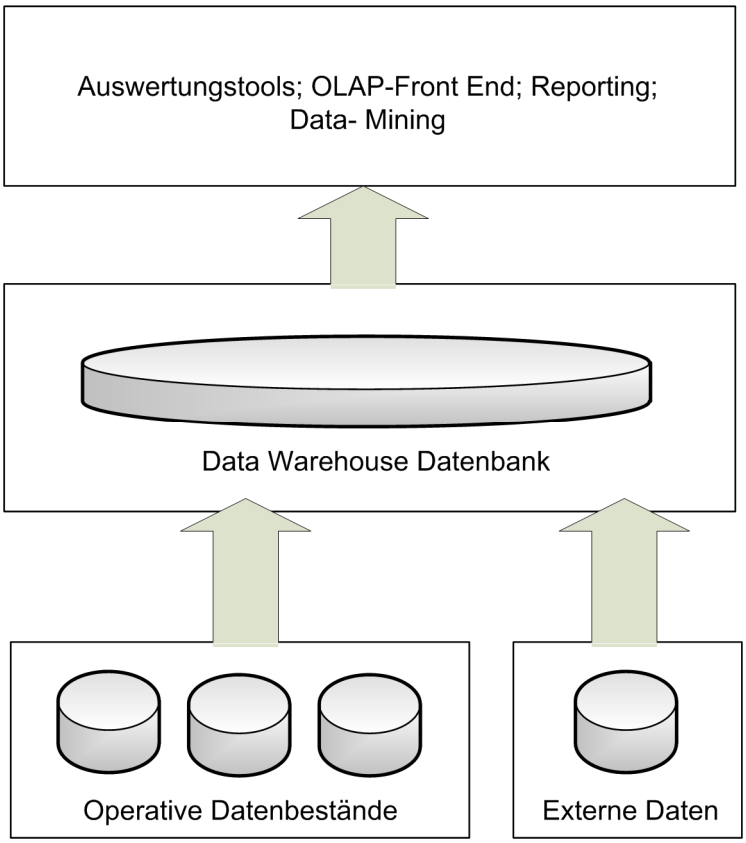

Abbildung 2 Data Warehouse Grundaufbau

Durch die Überführung der Informationen in ein Data Warehouse wird die Verarbeitung großer Datenmengen durch Analyse- und Auswertungstools effizienter. Die Produktivsysteme werden bei einer Auswertung oder Berichterstellung nicht belastet.

2.2.1 Merkmale eines Data Warehouse

Ein Data Warehouse zeichnet sich im Wesentlichen durch die Merkmale Themenorientierung, Zeitraumbezug, einheitlicher Struktur und Format der Daten sowie der Nicht-Volatilität aus[4].

Themenorientierung

Der Aufbau einer Data Warehouse Datenbank orientiert sich, anders als bei operativen Systemen, am Interessensgebiet der Management bzw. Führungsebene. Damit soll erreicht werden, dass für den Entscheidungsträger wichtige Informationen schnell und

[4] vgl. Inmon (2005), S. 29 ff

zuverlässig ausgegeben werden können. Häufig anzutreffende Interessensgebiete sind[5]

- Unternehmensstruktur (z.B. Geschäftsbereiche)
- Produktstruktur (z.B. Produktgruppen, Produkte)
- Regionalstruktur (z.B. Länder, Gebiete, Filialen)
- Zeitstruktur (z.B. Jahre, Quartale, Monate)

sowie Informationen zu den

- betriebswirtschaftliche Kennzahlen (z.B. Umsätze, Gewinn)
- und deren Ausprägung (Plan-, Ist-Werte)

Zeitraumbezug

Während in operativen Systemen meist nur der aktuell gültige Wert gespeichert ist, wird in einem Data Warehouse eine Historie der Daten angelegt. Mit gezielten Abfragen können Datenwerte der Vergangenheit ausgegeben werden, die zu diesem Zeitpunkt aktuell waren.

Eine Datenbank ohne periodische Bereinigung oder Archivierung kann mit der Zeit, in Abhängigkeit der periodisch hinzukommenden Datenmenge, eine nicht zu vernachlässigende Größe erreichen. In der Praxis sollte man sich eine Strategie erarbeiten, wie weit zurück Daten aus dem Data Warehouse benötigt und wie die älteren Daten archiviert werden.

Einheitliche Struktur und Format der Daten

Wichtig für eine Data Warehouse Datenhaltung ist das Überführen sämtlicher Daten aus den unterschiedlichen operativen Systemen in ein einheitliches widerspruchsfreies Format. Die meisten operativen Systeme haben eine proprietäre Datenhaltung und Struktur, was das Überführen in der Praxis äußerst schwierig werden lässt. Es muss für jedes System einzeln auf die im System vorhandenen Redundanzen, Inkonsistenzen und semantischen Widersprüche eingegangen werden.

[5] Kemper, Baars, Mehanna (2010) S. 20

Nicht-Volatilität

Unter Volatilität wird der Umfang, mit dem sich Daten während der normalen Nutzung ändern, bezeichnet. In operativen Systemen werden häufig Schreibvorgänge auf dem Datenbestand ausgeführt. Es findet also in kurzen Abständen eine Aktualisierung statt. In einem Data Warehouse werden für gewöhnlich die Informationen eingelesen und stehen danach für Leseprozesse zur Verfügung. Durch diesen Umstand sind Abfragen auf ein Data Warehouse zu jedem Zeitpunkt reproduzierbar.

2.2.2 Architekturvarianten

Unter der Architektur eines Data Warehouse versteht man den grundsätzlichen Aufbau der Datenhaltung des Data Warehouse. Gängige Architekturen sind ein zentrales Data Warehouse oder die Aufteilung des Datenbestandes auf verschiedene Data Marts in einem dezentralen DWH.

Unter einem Data Mart versteht man eine (Teil-)Kopie des Data Warehouse Datenbestands. Die Daten werden meist nach bestimmten Organisationsbereichen oder Aufgabenanforderungen in Data Marts aufgeteilt. Für die verschiedenen Anwendungsbereiche sind unterschiedliche Datenbankentwürfe notwendig, um eine erhöhte Performance zu erreichen oder Rechen- oder Netzwerklast zu minimieren.

Zentrales Data Warehouse

In einem zentralen Data Warehouse werden alle operativen und externen Daten in einem einzigen Datenbankmanagementsystem gespeichert. Sofern das zugrundeliegende Datenbank Management System Clustering oder die physikalische Verteilung der Daten auf unterschiedliche Server und Standorte erlaubt, kann es zur Steigerung der Performance und Ausfallsicherheit genutzt werden.

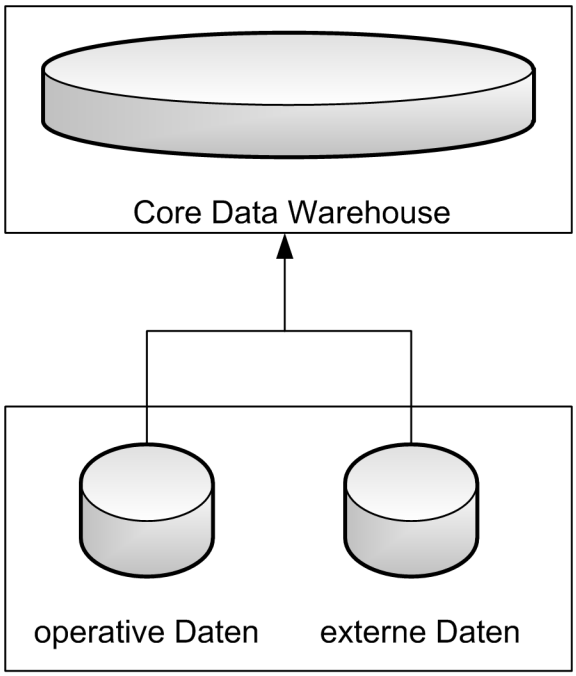

Abbildung 3 zentrale Data Warehouse Architektur

Das Erstellen einer einzigen, zentralen Data Warehouse Datenbank ist in den meisten praktischen Fällen eine nicht zu unterschätzende Herausforderung[6]. So müssen alle internen und externen Daten auf ein einheitliches Format gebracht werden, um in das Data Warehouse überführt zu werden. Die für das DWH zu verarbeitenden Datenmengen können schnell ansteigen und die Performance maßgeblich negativ beeinflussen. Aufgrund der genannten Nachteile wird in den meisten Fällen eine dezentrale Lösung bevorzugt[7].

Dezentrales Data Warehouse

In einem dezentralen Data Warehouse werden die Daten themenorientiert in unabhängige Data Marts überführt. Gewöhnlich wird für jedes Data Mart eine eigene, auf die Daten optimierte Datenbank verwendet. Häufig werden die Data Marts nach Fachabteilungen unterteilt.

Durch die Aufteilung in verschiedene Data Marts, die auf unterschiedlichen physikalischen Maschinen gehostet werden können, behindern sich gleichzeitige Auswertungen in zwei oder mehr Fachabteilungen nicht. Es findet also eine Lastverteilung statt.

[6] vgl. Kemper, Baars, Mehanna (2004) S.19
[7] vgl. Kemper, Baars, Mehanna (2004) S.20

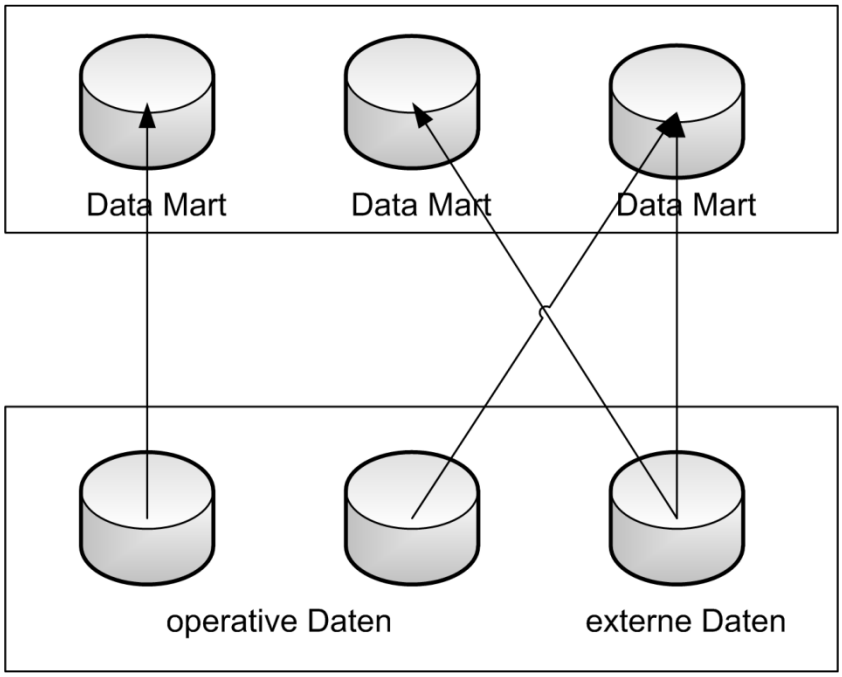

Abbildung 4 dezentrale Data Warehouse Architektur

Als Nachteil ist zu erwähnen, dass eben diese Splittung der Daten bei der Auswertung gesamtunternehmerischer Informationen Probleme hervorrufen kann. Für eine solche Abfrage müssen verschiedene Data Marts und Ihre Datenbankunterbauten verknüpft werden, was eine drastische Performanceeinbuße zur Folge haben kann.

Nabe-Speiche-Architektur

Bei einer Nabe-Speiche-Architektur (engl. Hub-and-Spoke) werden die Daten über einen ETL Prozess in das Data Warehouse und in den Operational Data Store (ODS) überführt. Der ODS stellt eine transaktionsorientierte Datenhaltung bereit und ähnelt der Datenhaltung der operativen Systeme stark. In ihm wird keine Historie gespeichert, jede neue Transaktion überschreibt die vorhandenen Daten.

Der ETL Prozess dient zur Überführung der Daten in die themenorientierte Struktur des Data Warehouses. ETL steht für *Extraction, Transformation and Loading*.

Das Core Data Warehouse stellt die Nabe und die Data Marts die Speichen dar.

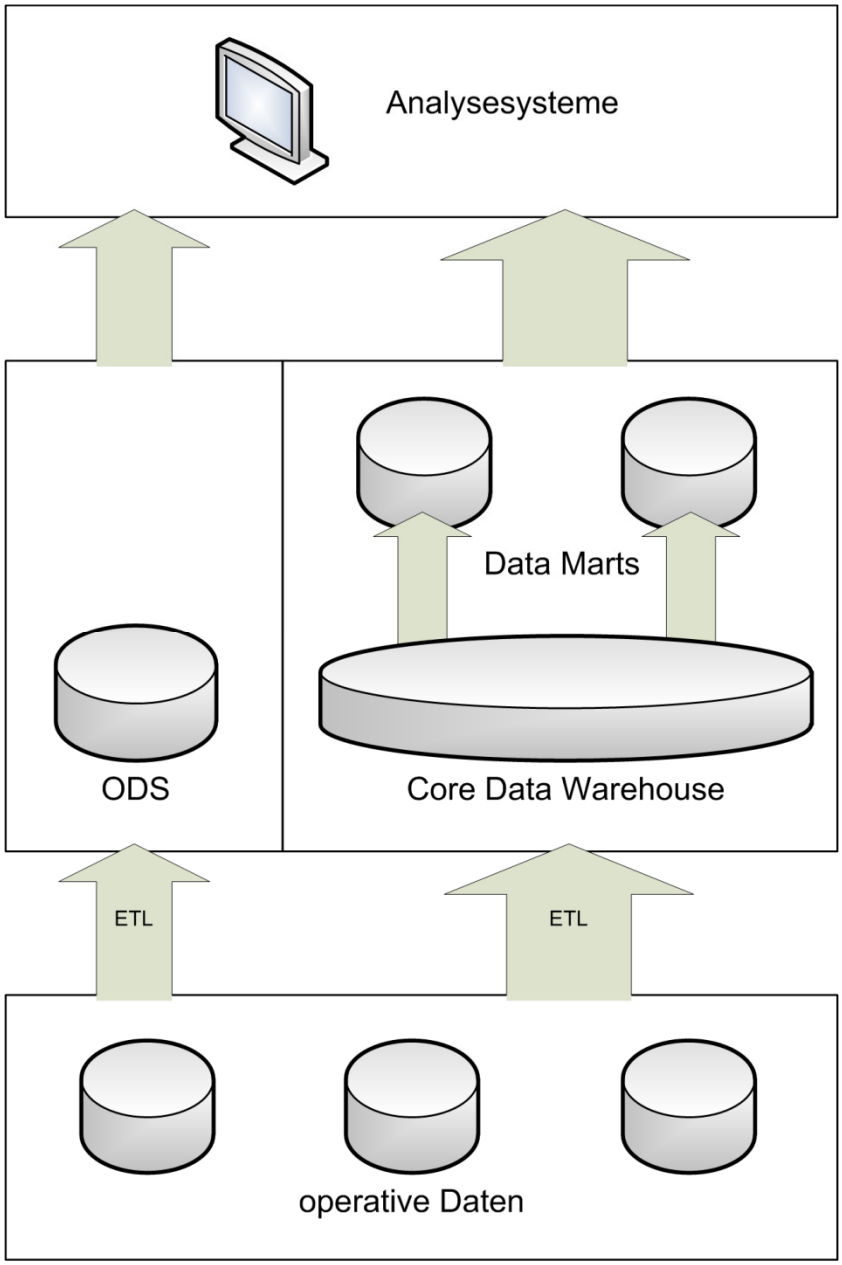

Abbildung 5 ODS erweiterte DWH-Architektur

An das Core Data Warehouse werden Data Marts angebunden, die die Daten aus dem Data Warehouse für ihren Anwendungsfall aufbereitet speichern.

2.3 On-Line Analytical Processing

Die Abkürzung OLAP (On-Line Analytical Processing) wurde erstmals in einer Veröffentlich von E.F. Codd, S.B. Codd und C.T. Salley im September 1993 verwendet. In dieser Veröffentlichung prüfen die Autoren, inwieweit herkömmliche, relationale Datenbanken mit den Anforderungen multidimensionaler analytischer Abfragen kompatibel sind. Unter einer multidimensionalen Analyse wird das Verdichten von Daten verstanden, um sie unterschiedlich darzustellen und eine Analyse zu ermöglichen[8].

Der Begriff OLAP wird weitestgehend als synonym für multidimensionale Datenanalyse verwendet, wobei sich OLAP als favorisierter Begriff durchgesetzt hat.

2.3.1 Grenzen relationaler Datenbanken

Für die Analyse werden, wie bereits angesprochen, multidimensionale Abfragen benötigt. Für eine solche Auswertung ist SQL nur bedingt nutzbar. Das kann zum Beispiel wie folgt beschrieben werden:

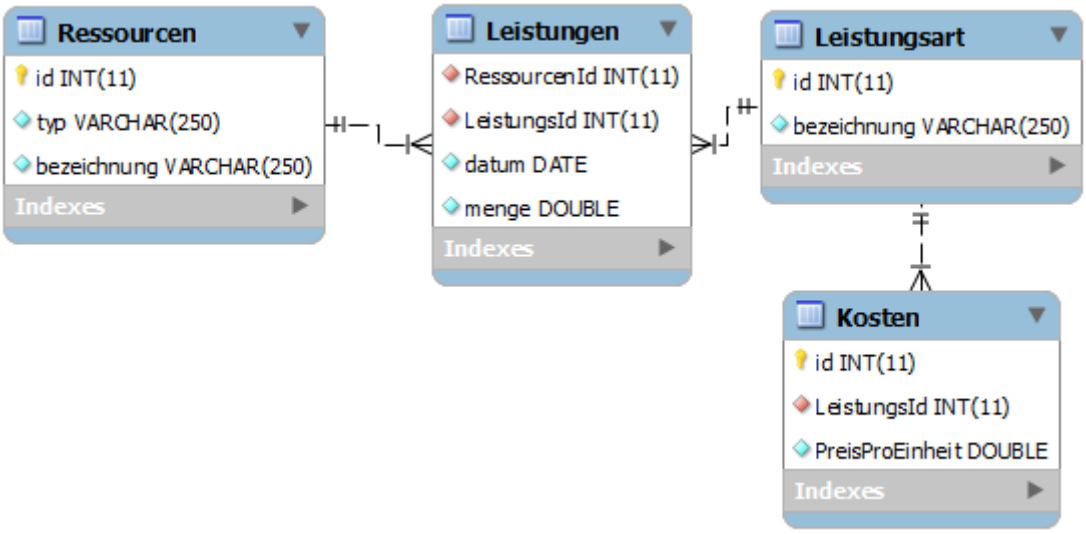

Abbildung 6 ERM Diagramm für SQL Beispiel

»Wie viele *Kilometer* Kanal wurden in den *vergangenen Jahren* mittels *Hochdruckspül- und Saugfahrzeugen gereinigt*?«

[8] vgl. Azevedo, Pedro, et al. (2006) S.38

Die Dimensionen dieses Beispiels sind der Zeitraum (vergangene Jahre), Leistungsart (Reinigung) und die verwendeten Ressourcen (Hochdruckspül- und Saugfahrzeuge). Der geforderte Fakt ist die Menge der gereinigten Kilometer.

Folgendes SQL-Statement gibt eine Übersicht nach Jahr, gereinigten Kilometer, Ressourcentyp und der Leistungsart aus:

```
1:  SELECT YEAR(datum) as Jahr, SUM(MENGE)/1000 as km, r.typ, la.bezeichnung as
    Leistung
2:  FROM Leistungen as l
3:  LEFT JOIN Ressourcen as r ON r.id = l.RessourcenId
4:  LEFT JOIN Leistungsart AS la ON la.id = l.LeistungsId
5:
6:  WHERE (typ = 'Hochdruckspülfahrzeug' OR typ = 'Saugfahrzeug')
7:          AND la.bezeichnung = 'Reinigung'
8:
9:  GROUP BY YEAR(datum), typ
10: ORDER BY Jahr DESC, km, typ
```

Code 1 SQL Abfrage für Beispiel #1

Die o.g. SQL-Abfrage erzeugt eine Ausgabe in der Form „Jahr, km, Typ, Ressource“ (siehe Abbildung 7). Die Ausgabe ist vollständig, aber wenig übersichtlich. Eine Ausgabe als PivotTable ist hier sinnvoll und zu bevorzugen.

Jahr	km	typ	Leistung
2012	0.905	Hochdruckspülfahrzeug	Reinigung
2012	0.927	Saugfahrzeug	Reinigung
2011	1.629	Hochdruckspülfahrzeug	Reinigung
2011	2.013	Saugfahrzeug	Reinigung
2010	1.938	Hochdruckspülfahrzeug	Reinigung
2010	1.979	Saugfahrzeug	Reinigung
2009	1.641	Saugfahrzeug	Reinigung
2009	1.924	Hochdruckspülfahrzeug	Reinigung
2008	1.688	Hochdruckspülfahrzeug	Reinigung
2008	1.807	Saugfahrzeug	Reinigung
2007	1.291	Hochdruckspülfahrzeug	Reinigung
2007	1.81	Saugfahrzeug	Reinigung
2006	1.528	Saugfahrzeug	Reinigung
2006	1.873	Hochdruckspülfahrzeug	Reinigung
2005	1.595	Hochdruckspülfahrzeug	Reinigung
2005	1.809	Saugfahrzeug	Reinigung

Abbildung 7 (gekürzte) Ausgabe von Code 1

In Abbildung 8 wird die Ausgabe von Abbildung 7 Abbildung 6in einer Excel-PivotTable dargestellt. Jedes der filterbaren Felder stellt eine Dimension dar. Es kann mit wenigen Klicks der Jahres- oder Leistungsfilter geändert werden. Die Daten der Tabelle werden automatisch aktualisiert.

1	Jahr	2011
2	Leistung	Reinigung
3		
4	**Ressource**	**Summe von km**
5	Hochdruckspülfahrzeug	1,629
6	Saugfahrzeug	2,013
7	**Gesamtergebnis**	**3,642**
8		

Abbildung 8 Excel-PivotTable

Die Daten für das Excel Beispiel werden von dem in Code 1 genannten SQL-Statement generiert. Für jede Filtermöglichkeit müsste das SQL-Statement bearbeitet werden und die Komplexität des Statements würde weiter steigen. Um weitere Auswertungen anzulegen, benötigt man geschultes Fachpersonal mit SQL Kenntnissen (und Berechtigungen).

Dieses Beispiel zeigt, dass weitere Werkzeuge nötig sind, um multidimensionale Daten für den Endanwender einfach auswertbar zu machen.

2.3.2 Der multidimensionale Entwurf (OLAP)

Dimension

Bei einer Datenanalyse in Unternehmen werden häufig aggregierte Daten benötigt. Analytische Fragestellungen der Managementebene des Unternehmens weichen gewöhnlich stark von denen aus einzelnen Geschäftsvorfällen ab.

Typische Fragestellungen sind zum Beispiel. »Wie viele *Kilometer* Kanal wurden in den vergangenen Jahren mittels Hochdruckspül- und Saugfahrzeugen *gereinigt*«.

Dimensionen ermöglichen eine unterschiedliche Sichtweise auf die Informationen. Sie können gruppiert und analysiert werden. Dimensionen bestehen aus Elementen. Elemente der Dimension Typ aus Abbildung 8 sind Hochdruckspülfahrzeug und Saugfahrzeug, das einzige angezeigte Element der Dimension Leistung ist hingegen die Reinigung.

Hierarchien

Bei einigen Dimensionen lassen sich die Elemente hierarchisch anordnen. Hierarchisch aufgebaute Dimensionen können sehr leicht als Baumdiagramme dargestellt werden. Eine geografische Dimension wäre zum Beispiel *Stadt -> Stadtteil -> Straße,* eine thematische *Abteilung -> Mitarbeiter*.

In einer multidimensionalen Analyse sollte es bis zu einem gewissen Punkt möglich sein, in eine detaillierte Ebene „hineinzuzoomen“ (Drill-Down) oder „herauszuzoomen“ (Drill-Up). Hierbei wird eine Hierarchieebene nach unten (detaillierter) oder nach oben (verdichteter) gegangen.

Kennzahlen

Unter Kennzahl, Messwert oder englisch Measures wird der Wert, den eine Dimension ausdrückt, bezeichnet. Jede multidimensionale Datenmenge muss aus mindestens einem Messwert bestehen. Kennzahlen einer multidimensionalen Datenmenge können zum Beispiel der Umsatz, Verbrauch oder die Kosten sein.

Cube

Über Dimensionen und Kennzahlen wird die multidimensionale Datenmenge definiert, die üblicherweise als Cube bezeichnet wird[9]. Der Anwender kann über das Auswählen der Achsen die Auswertung auf bestimmte Bereiche beschränken oder ausdehnen. Im Gegensatz zur Darstellung von herkömmlichen relationalen Datenbeständen lässt sich die Cube-Darstellung für den Anwender leichter interpretieren.

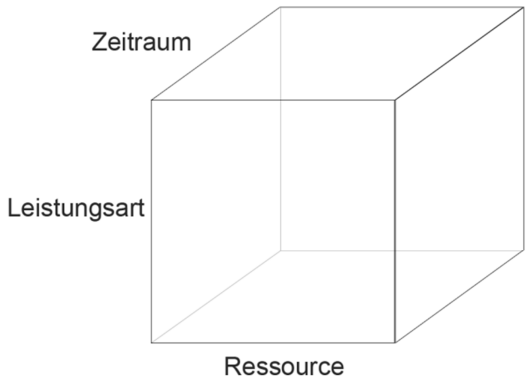

Abbildung 9 Dreidimensionale Datenmenge als Cube dargestellt

[9] Azevedo, Pedro, et al. (2006) S. 43

Auf einem OLAP-Cube können verschiedene Operationen angewandt werden, um eine andere Sicht auf die Informationen zu erhalten. Für gewöhnlich werden Operationen durch Mausklicks und Drag- und Drop-Aktionen auf den Cube angewandt, wobei zu beachten ist, dass sich das Verhalten zwischen den Herstellern unterscheiden kann[10].

Slicing und Dicing

Mittels den Slicing und Dicing Operationen kann die Ansicht auf die anzuzeigenden Informationen weiter eingeschränkt werden. Beim Slicing Verfahren wird der Cube durch die Reduktion einer Dimension in Scheiben „geschnitten“. Durch die Dicing-Operation wird hingegen ein kleiner Ausschnitt aus dem Cube angezeigt. Der Cube bleibt in diesem Fall multidimensional.

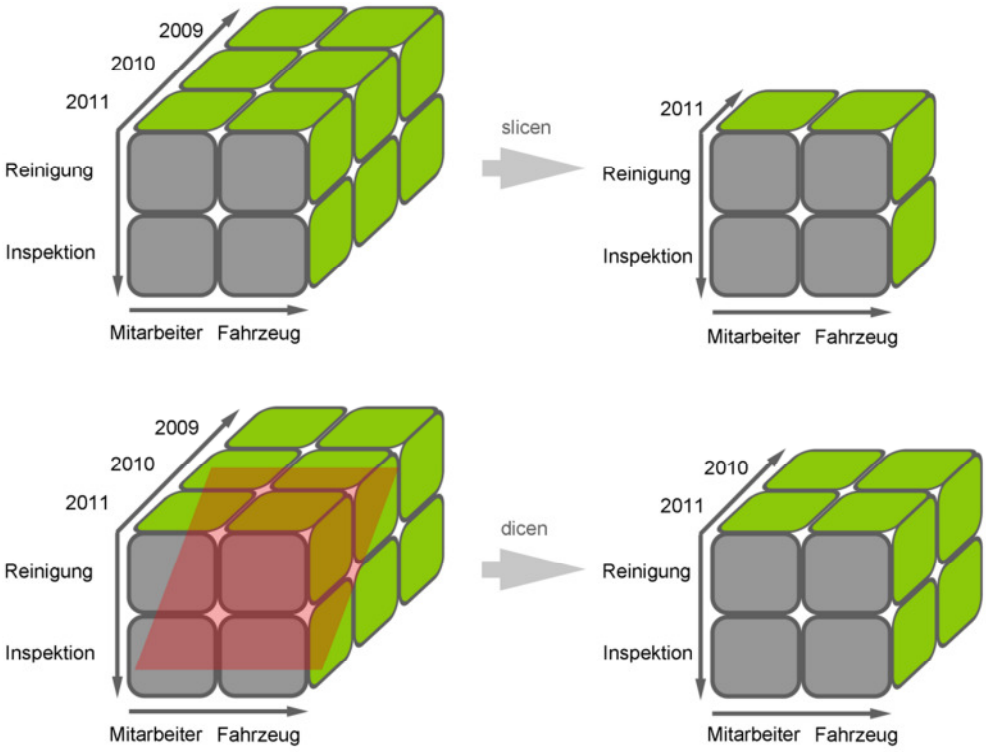

Abbildung 10 Slice/Dice Operation

Pivoting

Durch Pivoting (Rotation) wird der Cube an einer Achse gedreht und es wird mindestens eine andere Dimension sichtbar.

[10] Vgl. Thurnheer von Berneck (2003) S. 20

Drill-Down und Drill-Up

Die Drill-Down Operation ermöglicht eine detailliertere Darstellung des ausgewählten Bereichs (Hereinzoomen). Die Drill-Up Operation ermöglicht die Aggregation der Informationen (Herauszoomen).

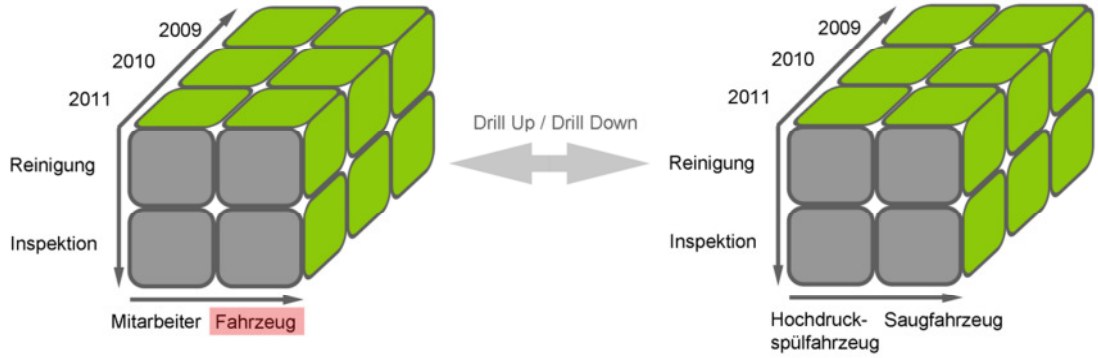

Abbildung 11 Drill Up / Drill Down Operationen

3 Übersicht existierender Lösungen

3.1 OLAP-Lösungen

OLAP Lösungen speichern Informationen in einer multidimensionalen Datenbank. Abfragen über eine multidimensionale Datenbank zeichnen sich in erster Linie durch ihre hohe Geschwindigkeit aus. Benötigt wird ein OLAP Server und ein Front End.

Die Front Ends können in zwei Kategorien eingeordnet werden. Es existieren Lösungen, in denen unerfahrene Anwender Analysen leicht erstellen können. Diese Lösungen sind aber stark in ihrer Flexibilität eingeschränkt, da eine Erstellung nur über die vorgegebenen Pfade möglich ist. Die zweite Kategorie umschließt die freien OLAP Front Ends. Sie sind komplexer in der Benutzerführung, bieten aber mehr Möglichkeiten, individuelle Analysen zu erstellen. Häufig werden Tabellenkalkulationsprogramme, allen voran Microsoft Excel, als freies OLAP Front End verwendet. Microsoft Excel kann über Add-Ins auf OLAP Datenbanken zugreifen[11].

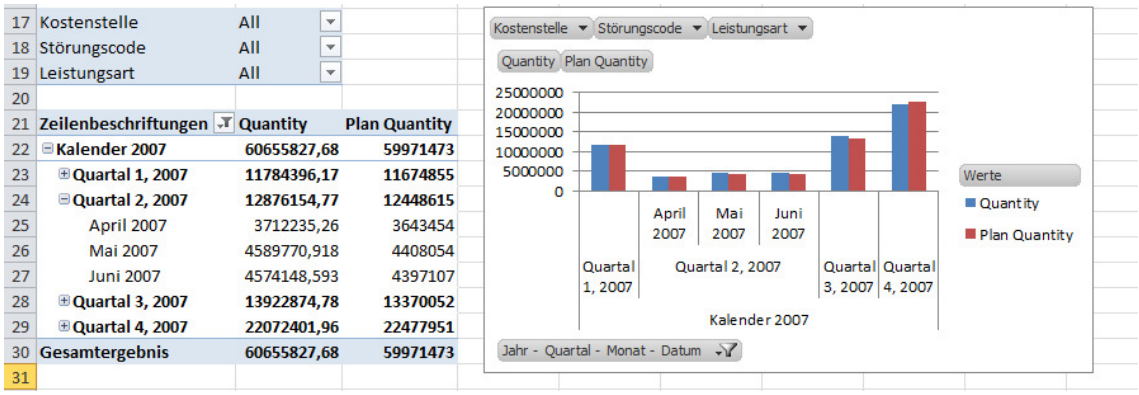

Abbildung 12 MS Excel Front End für Analysis Services

Als Server können freie Lösungen, wie Palo, oder auch kommerzielle Lösungen, wie die Microsoft Analysis Services genutzt werden.

[11] Vgl. Mahnart (2008)

	Palo	**MS Analysis Services**
Anbieter	Jedox AG Hauptsitz: Freiburg, Deutschland	Microsoft Inc., U.S. Hauptsitz: Redmond, United States
Datenintegration	Palo ETL Server Webbasierte GUI Quellen: JDBC ODBO/MDX	ETL über SSIS Entwicklungsumgebung BIDS Quellen: u.a. OLE DB, ODBC, XML, Excel MDX, DMX, ADOMD.NET, XMLA
Analyse	MOLAP In Memory Excel Add-In (Front End) Write Back	MOLAP, ROLAP, HOLAP Excel (Front End) Write Back
Reporting	Darstellung über Excel	Reporting Services
Schulungsaufwand *Excel erfahren* *Excel unerfahren*	 gering hoch	 gering hoch
Sonstiges	Basisserver GNU GPL Lizenz; Enterprise Version erhältlich	Gehört zum Funktionsumfang eines Microsoft SQL Servers

Tabelle 1 Vergleich OLAP Anwendung Palo vs. MS Analysis Services

3.2 Ad-Hoc Reporting-Lösungen

Mit Ad-Hoc Reporting-Lösungen erhält der Anwender ein Werkzeug, mit dem er selbstständig aus den produktiven Daten Berichte erstellen kann. Er benötigt dazu Zugriff auf die Datenbank der produktiven Systeme, wie zum Beispiel Dynamics NAV. Eine Ad-Hoc Lösung ist in den meisten Fällen sehr langsam, siehe Abbildung 36. Zusätzlich werden die produktiven Server einer starken Last ausgesetzt. Das kann in extremen Fällen zu Ausfällen der genannten Systeme beitragen.

Ein reines Ad-Hoc Reporting ist nur bei geringen Datenmengen und wenigen Beziehungen zu empfehlen und sollte vorab getestet werden. Die Ad-Hoc Reportierung der Arbeitsaufträge, siehe Kapitel 5, ist nicht zu empfehlen. Bei solchen umfangreichen Datenbeständen sollte über die Entwicklung eines Data Warehouse nachgedacht werden.

Ein großer Vorteil von Ad-Hoc Reports ist, dass der Entscheider die benötigten Informationen selbstständig aus den operativen Daten auswählen und meist über eine intuitiv bedienbare Oberfläche per Drag-and-Drop dem Bericht hinzufügen kann.

Ad-Hoc Reports können zum Beispiel mit FASTREPORT®.NET oder mit den Microsoft SQL Server Reporting Services erstellt und genutzt werden.

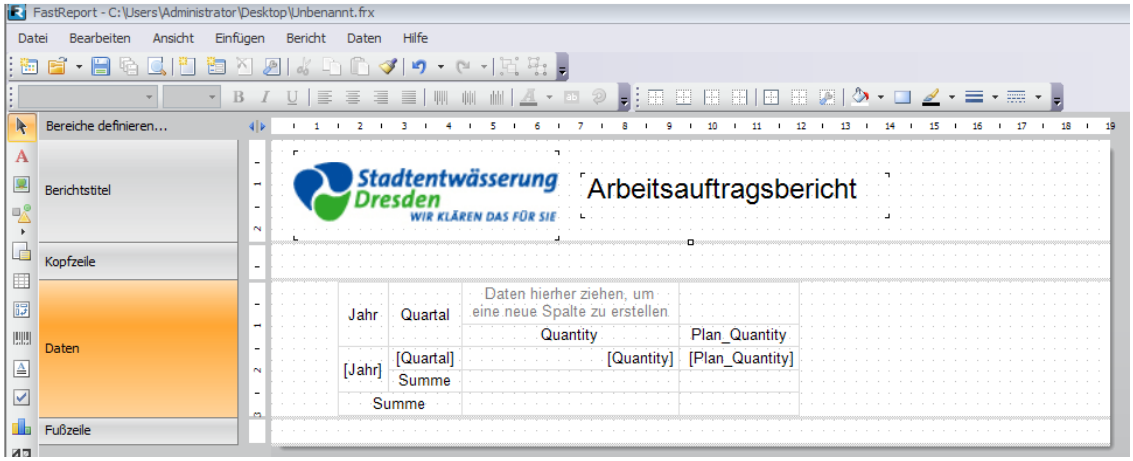

Abbildung 13 FASTREPORT®.NET Designer

	FASTREPORT®.NET	**MS SQL Server Reporting Services**
Anbieter	Fast Reports Inc., Russland Hauptsitz: Rostow am Don, Russland	Microsoft Inc., U.S. Hauptsitz: Redmond, United States
Quelldaten	MS Access, MS SQL Server, XML, OLE DB, ODBC	u.a. OLE DB, ODBC, XML, Excel, Flatfile
Scripting	.NET C#	VB, C#

Client/Server	Über zusätzlichen FASTREPORT®.NET SERVER	Sharepoint Anbindung, Reporting Services
Schulungsaufwand …. VS erfahren …. VS unerfahren	SQL Kenntnisse vorteilhaft mittel hoch	SQL Kenntnisse vorteilhaft mittel hoch

Tabelle 2 Vergleich Ad-Hoc Reporting Fastreport vs. MS Reporting Services

3.3 Business Intelligence Suites

BI Suites sind umfassende Softwarepakete und enthalten neben den Front End Werkzeugen auch eine Datenhaltung. Für eine schnelle Auswertungsperformance müssen die Daten in geeigneter Form vorgehalten werden. Das Front End greift nur auf die Daten in einer OLAP oder DataWarehouse Datenbank zu. Das operative System wird nur zur Aktualisierung der Datenbank abgefragt. Die Häufigkeit der Abfrage hängt von der benötigten Aktualität der Daten ab.

Bekannte Anbieter von BI Suites sind unter anderem IBM, Oracle und Microsoft.

	IBM Cognos Business Intelligence	**Microsoft Analysis and Reporting Services**
Anbieter	IBM Corp., U.S. Hauptsitz: Armonk, United States	Microsoft Inc., U.S. Hauptsitz: Redmond, United States
Dateningetration	ETL über Cognos DecisionStream Quellen: u.a. OLE DB, XML, ODBC MDX; XBRL	ETL über SSIS Entwicklungsumgebung BIDS Quellen: u.a. OLE DB, ODBC, XML, Excel MDX, DMX, ADOMD.NET, XMLA

Analyse	IBM Cognos Content Analytics Web Front End MS Excel Front End	Analysis Services MOLAP, ROLAP, HOLAP OLAP Front End MS Excel
Reporting	IBM Cognos Query and Reporting	Reporting Services Erstellung über BIDS
Data Mining	als Zusatzmodul	vorhanden
Schulungsaufwand …. MS Office erfahren …. MS Office unerfahren	 hoch hoch	 gering hoch

Tabelle 3
Vergleich BI Suites IBM Cognos Business Intelligence vs. MS Analysis and Reporting Services

4 Bedarfs- und Ist-Analyse

Vor der Umsetzung eines BI-Projektes ist es wichtig, gewisse Eckdaten zu definieren. Für die Bedarfs- und Ist-Analyse haben sich einige Kernfragen etabliert, anhand derer der Aufbau der BI-Lösung erfolgen kann.

4.1 Bedarfsanalyse[12]

- Wo gibt es Prozessprobleme / hohen manuellen Aufwand?
- Wo gibt es Probleme mit der Qualität der Daten?
- Welche Kennzahlen werden benötigt?
- Welche Dimensionen und Datentiefen werden benötigt?
- Welche Analyseperspektiven werden verwendet?
- Welche Aggregatsebene und Aktualität der Daten werden benötigt?

4.2 Ist-Analyse[13]

- Wo werden bereits BI-Methoden angewendet?
- Werden bereits Reporting Systeme, Data Warehouse Systeme, o.ä. verwendet?
- Wo liegen die Daten der operativen Systeme?
- Werden bereits Technologien verwendet, die eine BI Funktionalität bieten?
- Welches Know-how und welche personelle Ressourcen sind verfügbar?

[12] Vgl. Alexander (2010) S. 65
[13] Vgl. Alexander (2010) S. 65ff

4.3 Ist-Analyse am Beispiel der Stadtentwässerung Dresden GmbH

4.3.1 Analyse

- Wo werden bereits BI-Methoden angewandt?

  Für verschiedene Controlling Berichte werden bereits Reports erstellt. Sie werden meist direkt aus den produktiven Daten erstellt und haben so eine relativ lange Erstellungsdauer.

- Werden bereits Reporting Systeme, Data Warehouse Systeme o.ä. verwendet?

  Vorranging wird bei der Stadtentwässerung Dresden GmbH der Reporting Service des Microsoft SQL Servers genutzt. Eine Datawarehouse Datenbank wird genutzt, um externe Daten für den Bericht bereitzustellen.

- Wo liegen die Daten der operativen Systeme?

  Die benötigten internen Daten befinden sich zum größten Teil in MS Dynamics NAV und werden auf einem MS SQL Server gespeichert.

- Werden bereits Technologien verwendet, die eine BI Funktionalität bieten?

  Es wird bereits der Reporting Service von MS SQL Server verwendet.

- Welches Knowhow und welche personelle Ressourcen sind verfügbar?

  Die EDV-Abteilung der Stadtentwässerung Dresden GmbH verwaltet und administriert seit vielen Jahren MS SQL Server und hat ein entsprechendes Knowhow in der Verwaltung und Administration dieser Server. Kenntnisse in der Datenbankentwicklung und Abfragenerstellung sind vorhanden.

4.3.2 Zusammenfassung und Fazit

Die Mitarbeiter der EDV-Abteilung sind bereits im Umgang mit Microsoft SQL Servern geschult. Benötigte Lizenzen für eine erweiterte Nutzung der SQL Server sind ebenfalls bereits vorhanden. Das Einführen einer weiteren Software – unabhängig ob Desktop Edition oder Client/Server Architektur – ist nicht ohne hohen finanziellen und personellen Aufwand möglich. Außerdem werden bereits Microsoft Lösungen für das

Intranet eingesetzt (u.a. Sharepoint, IIS-Sites), was einen gewissen Wiedererkennungswert darstellt und eine Implementation erleichtert.

Aus diesen Gründen wird sich die weitere Arbeit mit dem Aufbau des Prototyps im MS Analytical/Reporting Services Universum beschäftigen.

5 Entwurf mit SQL Reporting Services

In diesem Kapitel wird der Entwurf eines prototypischen Aufbaus eines Reports am Beispiel der Stadtentwässerung Dresden GmbH gezeigt. Vor der Erstellung und Veröffentlichung des eigentlichen Reports müssen einige Vorüberlegungen erarbeitet und Entscheidungen getroffen werden.

Es muss sich mit dem Aufbau der „Roh“-Daten auseinander gesetzt werden, ein Data Warehouse errichtet und die Daten aus den produktiven Systemen in das DWH durch ETL-Prozesse übertragen werden.

Microsoft bietet mit den MS SQL Analysis und Reporting Services ein umfangreiches Werkzeug für den Aufbau eines Data Warehouse, OLAP Datenhaltung und die Generierung von Reports an.

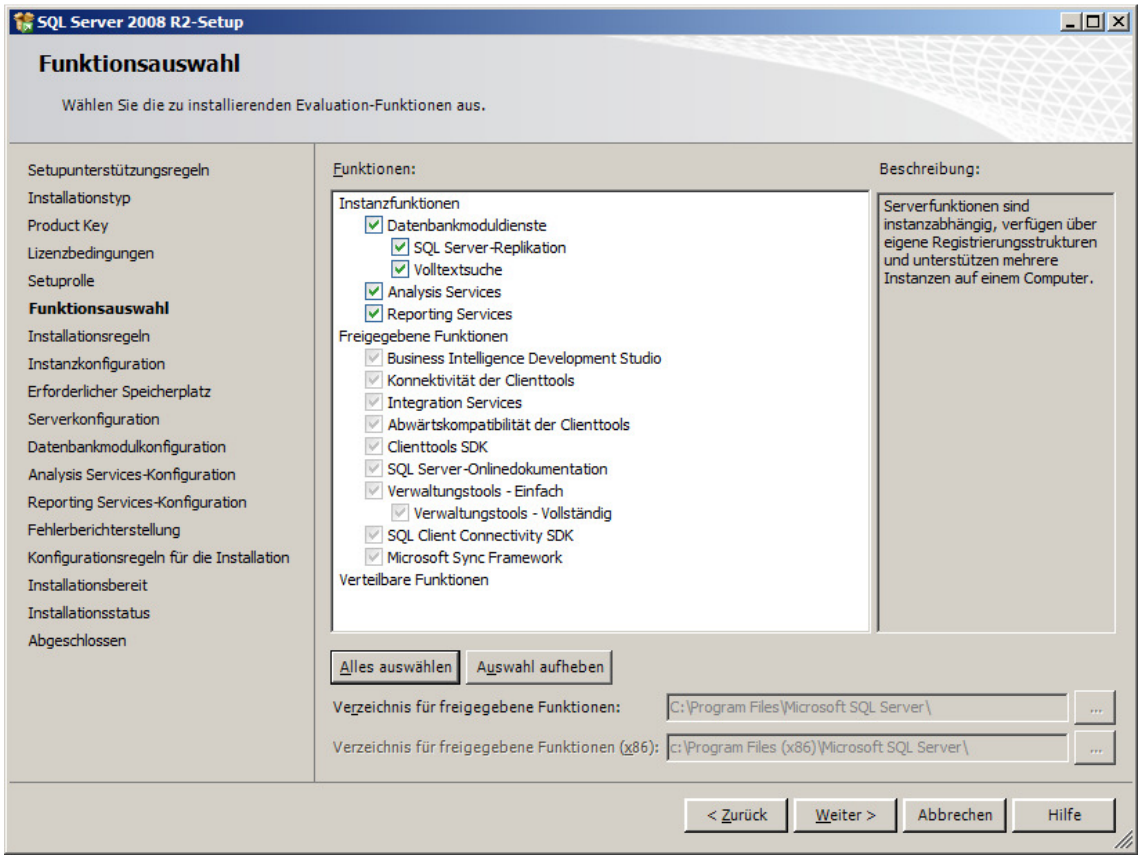

Abbildung 14 Funktionsauswahl Installation SQL Server 2008 R2

Die Analysis und Reporting Services sind Bestandteil der MS SQL Server ab der Standard-Edition und können über den Installationsmanager auch vorhandenen SQL Server Instanzen hinzugefügt werden.

5.1 Vorüberlegungen

Berichte sollten, wie in den vorangegangenen Kapiteln bereits erläutert, nicht direkt aus den operativen Datenbeständen, sondern aus einem Data Warehouse oder einer OLAP-Datenhaltung erstellt werden.

Vorteile dieser Methode sind eine erhöhte Performance der Berichtserstellung und eine Last-Reduzierung auf den produktiven Systemen. Nachteilig ist, dass durch eine weitere Speicherung der Daten ein erhöhter Bedarf an Speicherplatz anfällt. Um diesen Speicherbedarf so gering wie möglich zu halten und zusätzlich die Abfrageleistung zu erhöhen, müssen die Daten für den jeweiligen Einsatzweck aufbereitet gespeichert werden.

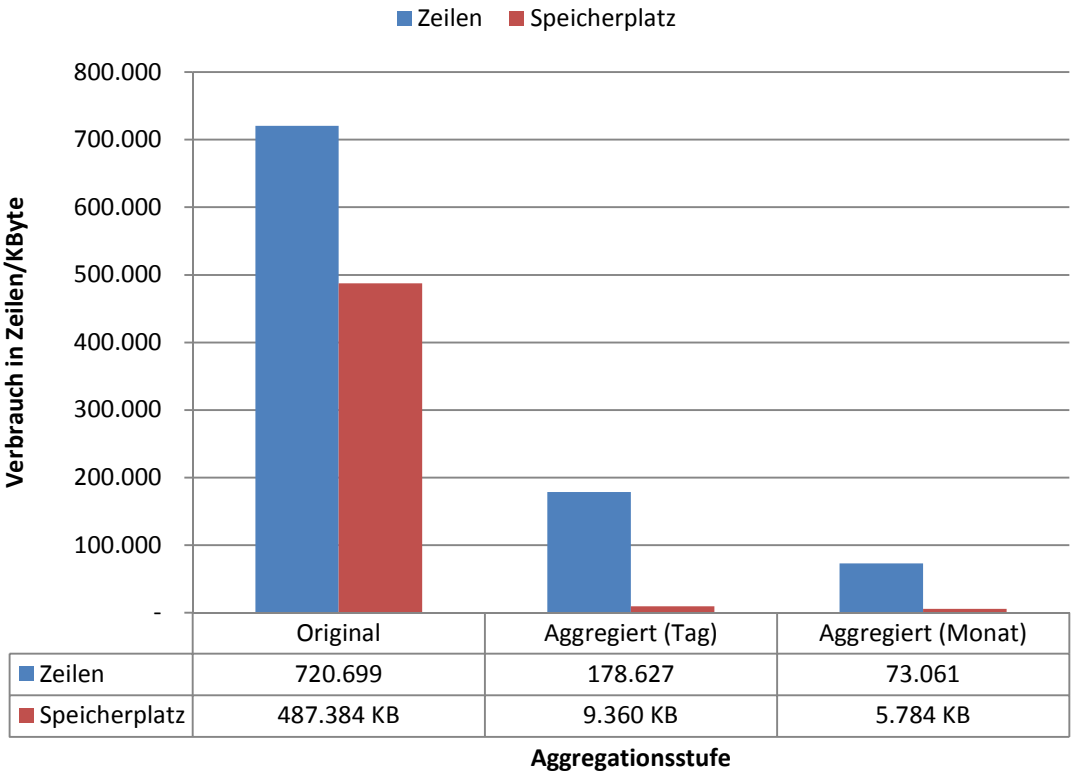

Abbildung 15 Speicherverbrauch nach Aggregationsstufen der Work Order Ledger Entry Tabelle

Der hier vorgestellte Bericht stellt eine Übersicht über die abgerechneten Arbeitsaufträge dar. Die benötigten Informationen werden zum einen aus MS Dynamics NAV und zum anderen aus einer separaten Planungs-DB exportiert.

Die für den Bericht benötigten Informationen müssen aus Navision und einer zusätzlichen „Planungs“-Datenbank extrahiert werden.

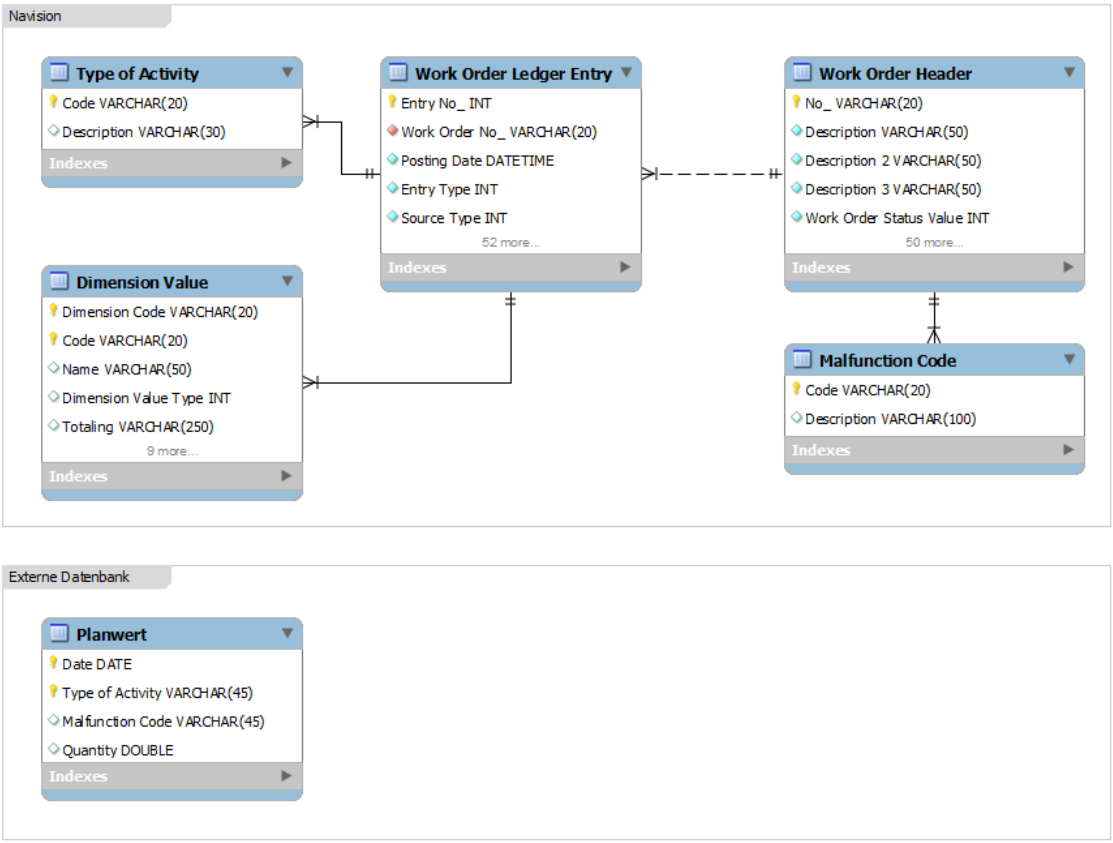

Abbildung 16 (gekürztes) ERM der operativen Daten

- Wo gibt es Prozessprobleme / hohen manuellen Aufwand?

Die Erstellung der existierenden Berichte haben eine sehr lange Laufzeit. Sie werden direkt aus den produktiven Datenbeständen generiert und liegen in einer ungünstigen Form für die Berichtsgenerierung vor.

- Welche Kennzahlen werden benötigt?

Die für den Bericht benötigten Kennzahlen sind die Quantity der Arbeitsaufträge, sowie die Quantity der Planung.

- Welche Dimensionen werden benötigt?

Der spätere Bericht soll über die Dimensionen Leistungsart (Type of Activity), Zeit (Posting Date), Kostenstelle (Global Dimension 1 Code) und den Störungscodes (Malfunction Code) verfügen.

- Welche Aggregatsebene und Aktualität der Daten werden benötigt?

Die kleinste benötigte Zeiteinheit ist »Monat«. Die Informationen können während des ETL-Prozesses auf den Monat in Abhängigkeit von Type of Activity, Global Dimension 1 Code und des Malfunction Codes aggregiert werden. Die Planwerte befinden sich bereits in der benötigten Aggregation (Monat).

5.2 Data Warehouse Entwurf

Das DataWarehouse speichert die aufbereiteten Informationen der Produktivsysteme. Für diesen Bericht werden Daten aus zwei unterschiedlichen Datenbanken benötigt und müssen zusammengeführt gespeichert werden.

Für den ETL-Prozess bietet der MS SQL Server die SQL Server Integration Services an. Mit den SSIS werden dem Entwickler Werkzeuge zur Verfügung gestellt mit denen er Daten aus den verschiedensten Datenquellen (u.a. MS Excel Dateien, div. Datenbanksysteme, Flatfiles, XML-Dateien) auslesen, transformieren und in einem Data Warehouse abspeichern kann.

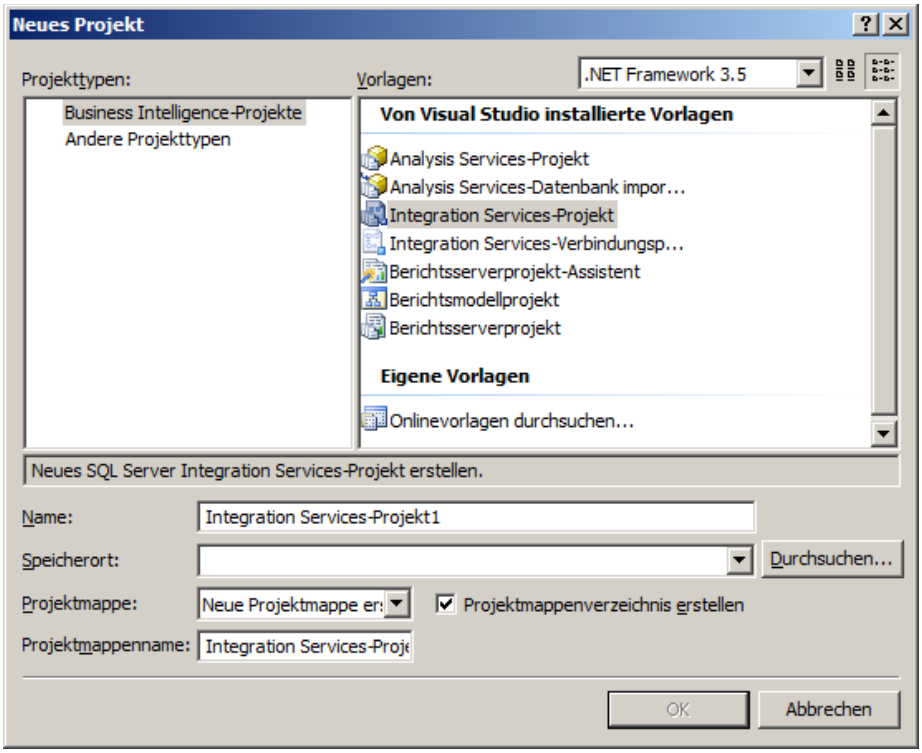

Abbildung 17 Neues SSIS Projekt in VS anlegen

Die Modellierung des ETL-Prozesses findet im Business Intelligence Development Studio, einem Visual Studio Add-On, statt. Das BIDS kann aus einem Visual Studio Professionell heraus gestartet oder mit der Installation der Analysis und Reporting Services des SQL Servers mit installiert werden.

5.2.1 Quelldaten

Die Quelldaten befinden sich in zwei unterschiedlichen Datenbanken. Die Work Order Header und Work Order Ledger Entry Tabellen sind Teile der Navision Datenbank. Die Planwert Tabelle wird manuell in einer separaten Datenbank gefüllt. Die Informationen der Quelldatenbanken können sich jederzeit, auch für weit zurückliegende Einträge, ändern.

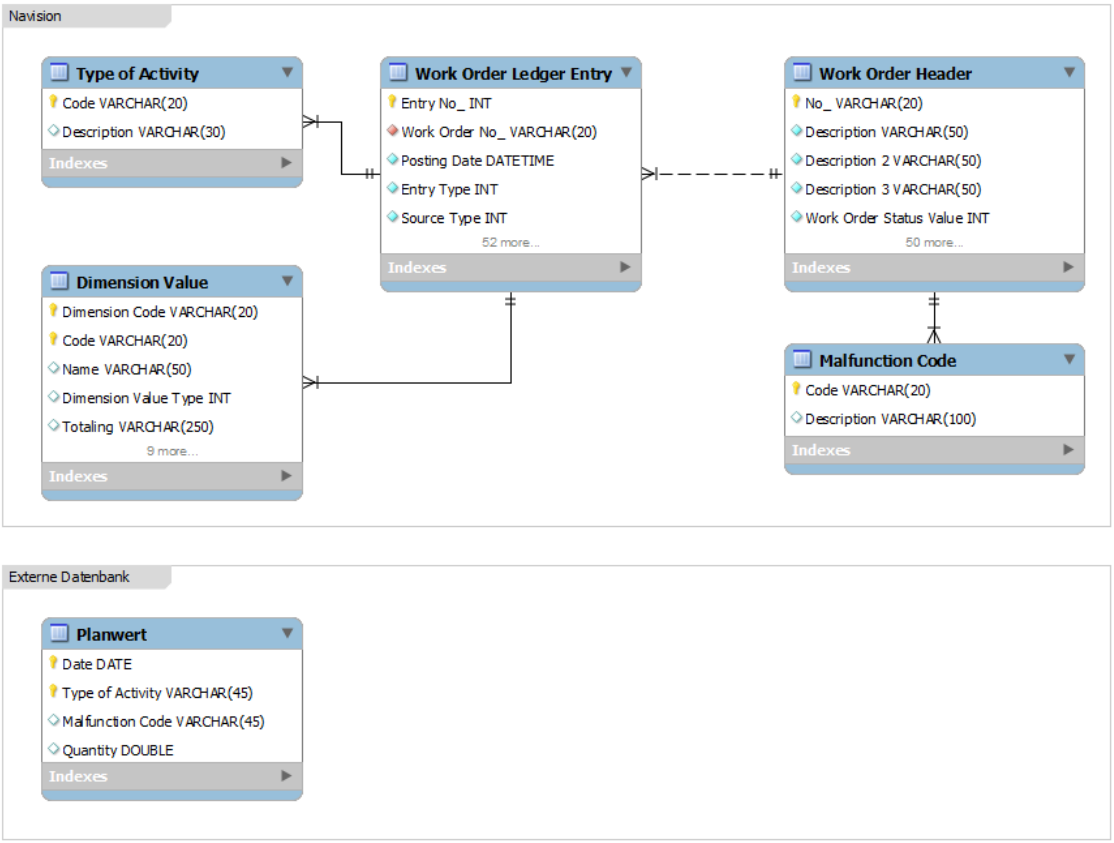

Abbildung 18 (gekürztes) ERM der operativen Daten

Work Order Header und Work Order Ledger Entry

Die Work Order Header Tabelle dient als Kopfzeile für die Arbeitsaufträge. In ihr befinden sich Informationen wie zum Beispiel die Beschreibung des Auftrages, Störungscode, geplante Fertigstellung usw. In der Work Order Ledger Entry Tabelle befinden sich die Informationen für die einzelnen Arbeitsaufträge. Für den Bericht werden die Zeit, Leistungsart, Kostenstelle, Störungscode und die Anzahl der Stunden benötigt.

Planwert

Die Planwertdaten werden manuell von den betreffenden Abteilungen über ein Webfrontend eingetragen. Über die Felder Date_Key, Type of Activity und Malfunction Code werden die Planwertdaten mit den Produktivdaten verknüpft. Die Kostenstelle spielt bei den Planwerten keine Rolle. In diesem Beispiel wird die Menge in gleichen Teilen auf die Kostenstellen aufgeteilt.

Type of Activity, Dimension Value und Malfunction Code

Die Tabellen Type of Activity, Dimension Value und Malfunction Code stellen die Dimensionen der späteren Abfragen dar. In ihnen sind die Bezeichnungen der Keys gespeichert. Für eine für den Nutzer lesbarere Form der Auswertung werden im späteren Report die Bezeichnungen anstatt nur die Fremdschlüssel angezeigt.

5.2.2 Zieldatenbank

Die Zieldatenbank besteht aus einer Faktentabelle und drei Dimensionstabellen. In die Faktentabelle werden die Kennzahlen Menge und Planungsmenge in Verknüpfung mit den Fremdschlüsseln der Dimensionen monatsgenau überführt. In den Dimensionstabellen werden der Primärschlüssel und die Bezeichnung abgelegt.

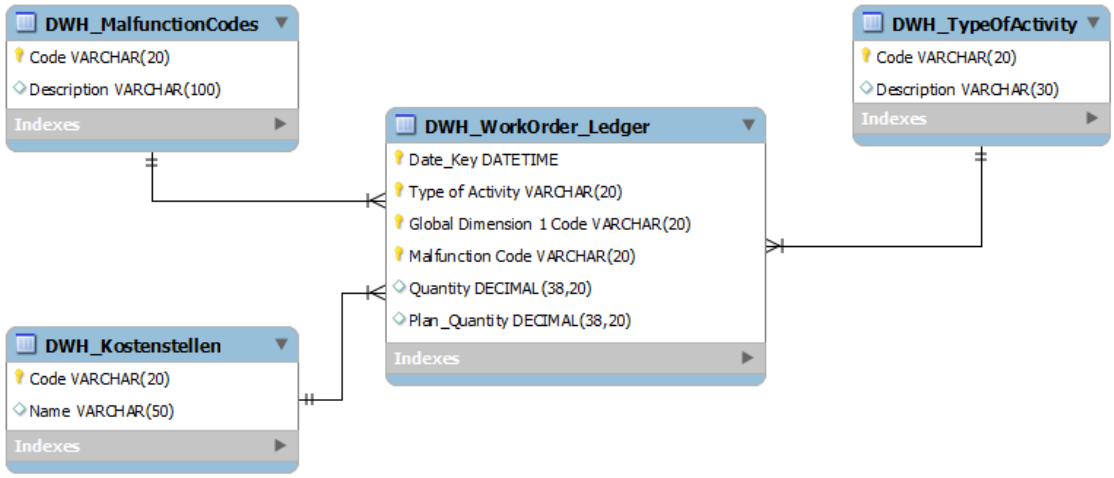

Abbildung 19 Ziel Datenbank

Eine Zeitdimension mit hierarchischer Struktur kann mit dem BIDS bei der Erstellung von OLAP-Cubes automatisiert als Servertabelle erstellt werden. Eine manuelle Erstellung im Vorfeld ist nicht notwendig.

Das in Abbildung 19 dargestellte ERM der Zieldatenbank stellt ein Sternschema dar. Eine Faktentabelle ist verbunden mit einer Vielzahl an Dimensionstabellen. Ein Sternschema ist neben dem Snowflake Schema (Schneeflocken Schema) ein bevorzugtes Quelldatenbankschema für die Erstellung von OLAP-Cubes.

5.2.3 ETL-Prozess

Der ETL-Prozess wird über das Business Intelligence Development Studio erstellt. Im ersten Schritt werden alle Zieltabellen geleert, da sich die produktiv Daten auch vergangener Zeiträume ändern können.

Nachdem die vorhandenen Tabellen erfolgreich geleert wurden, werden die Work Order Header und Ledger Entry Informationen geladen, aggregiert und in die DWH-Tabelle geschrieben.

Im Anschluss werden die Planwertdaten geladen und ihren Work Order Ledger Entries zugeordnet.

Bevor die Übertragung der Informationen der Quelldaten an den SSIS erfolgt, wird die Tabelle des Data Warehouse geleert.

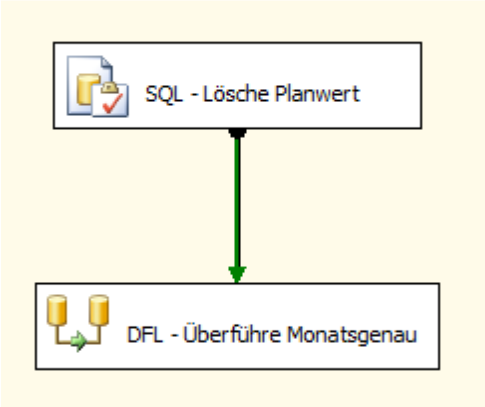

Abbildung 20 Reihenfolge Ablaufsteuerung (gekürzt)

Bei einer erfolgreichen Leerung der Zieltabelle werden ein neuer Datenfluss in den Ablaufplan eingefügt und die MS Navision Zugangsdaten als Ziel-Datenbank eingetragen.

Work Order Header / Work Order Ledger Entry

Für den ETL-Prozess muss ein Datenfluss in der Ablaufsteuerung definiert werden. In einem Datenfluss können verschiedene Elemente aus der Datenflussquelle, Datenflusstransformation und Datenflussziel Toolbox aneinander gereiht werden. Die Abarbeitung erfolgt sequenziell.

Für die Transformation der Daten gibt es eine Vielzahl von Möglichkeiten. Die Informationen können per „Aggregat"-Werkzeug, SQL-Statement, VB/C# Programmierung und vielen anderen Werkzeugen aufbereitet werden.

Für den Vergleich der resultierenden Größe der aggregierten Tabellen wurde für die tagesgenaue Tabelle das Aggregat-Werkzeug des Visual Studios und für die monatsgenaue Tabelle ein SQL-Statement verwendet. Für ein und dasselbe Resultat können unterschiedliche Werkzeuge oder Kombination aus diesen genutzt werden. Die Wahl der Werkzeuge obliegt in den meisten Fällen dem Entwickler.

Aggregation tagesgenauer Arbeitsaufträge

Für die Aggregation der Daten über das Aggregationswerkzeug wird über eine OLE-DB Quelle die komplette produktiv Tabelle – gejoint mit den Headerdaten – eingelesen und an das Aggregationswerkzeug weitergeleitet.

```
1: SELECT [Posting Date], [Type of Activity], Quantity, [Global Dimension 1
   Code], [Malfunction Code]
2: FROM dbo.[CRONUS TEST SE DD GmbH$Work Order Ledger Entry] as wole
3: LEFT JOIN dbo.[CRONUS TEST SE DD GmbH$Work Order Header] as woh ON
   wole.[Work Order No_] = woh.[No_]
```

Code 2 "Extract" des kompletten Datenbestandes

Über ein grafisches Frontend können alle Spalten, die aggregiert werden müssen, über Dropdown-Listen sowie der gewünschte Vorgang ausgewählt werden.

Das Frontend ähnelt dem Access-Abfragen-Generator und kann mit grundlegenden SQL bzw. Datenbankkenntnissen konfiguriert werden.

Um eine spätere Filterung nach Datum, Leistungsart, Kostenstelle und Störungscode zu ermöglichen, müssen diese Felder gruppiert werden. Die für den jeweiligen Einsatz aufgebrachten Stunden (Quantity) werden summiert.

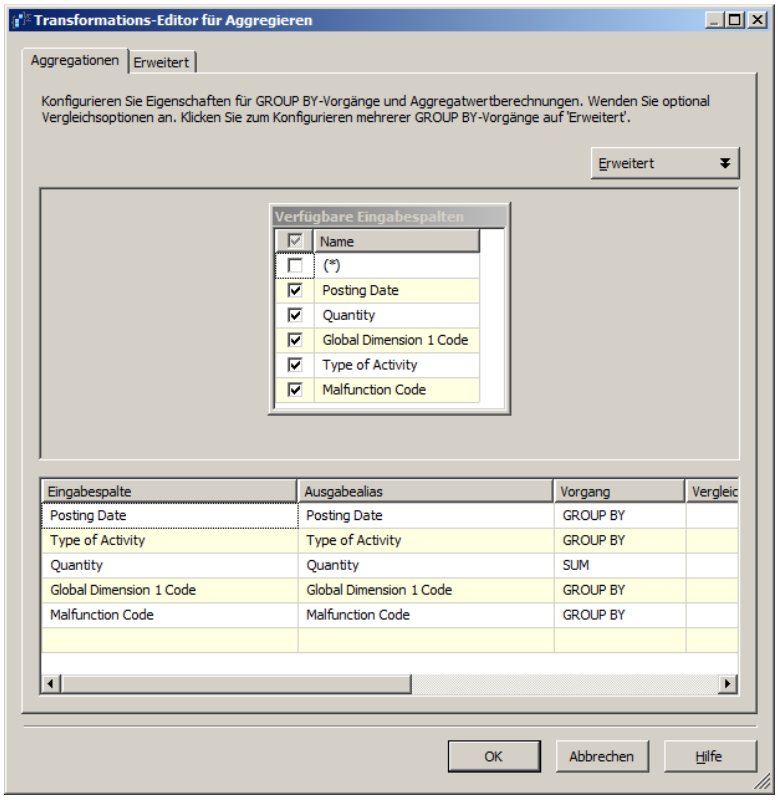

Abbildung 21 Transformations-Editor für das Aggregieren

Die transformierten Daten werden, nachdem sie von dem Transformationsprozess verarbeitet wurden, an eine OLE-DB Zielverbindung weitergeleitet. Die Zielverbindung schreibt die Daten im Anschluss in die *DWH_WorkOrderLedger_Day* Tabelle.

Aggregation monatsgenauer Arbeitsaufträge

Die Aggregation der monatsgenauen Daten wird mittels SQL auf dem Datenbankserver vorgenommen. Hierzu wird, anders als bei der tagesgenauen Aggregierung, das Extract-Statement um die benötigten Aggregationsfunktionen erweitert.

Der Transformationsprozess wird vollständig auf den produktiven Datenbankserver ausgelagert. Die Aggregation wird für das Jahr und den Monat vorgenommen. Um das zu erreichen muss das Datum zuvor in Jahr und Monat aufgeteilt werden. Für die Monatsdarstellung in der DWH-Tabelle wurde der 1. jeden Monats gewählt, um weiterhin mit den Date-Typen des SQL Server arbeiten zu können.

```
1: SELECT CAST(YEAR(wole.[Posting Date]) AS varchar(20)) + '-' +
   CAST(MONTH(wole.[Posting Date]) AS varchar(20)) + '-1' AS [Posting Month],
   wole.[Type of Activity], SUM(wole.Quantity) AS Sum_Quantity, wole.[Global
   Dimension 1 Code], [Malfunction Code]
2: FROM dbo.[CRONUS TEST SE DD GmbH$Work Order Ledger Entry] AS wole LEFT
   OUTER JOIN
```

```
3:     dbo.[CRONUS TEST SE DD GmbH$Work Order Header] AS woh
   ON wole.[Work Order No_] = woh.No_
4: GROUP BY YEAR(wole.[Posting Date]), MONTH(wole.[Posting Date]),
   wole.[Type of Activity], wole.[Global Dimension 1 Code], [Malfunction Code]
```

Code 3 Extract/Transform Monatsgenauer Arbeitsaufträge

Die hier neu erstellten Spalten müssen im Ziel-Editor den Zielspalten zugeordnet werden. Das kann über den Ziel-Editor der OLE-DB Verbindung erfolgen.

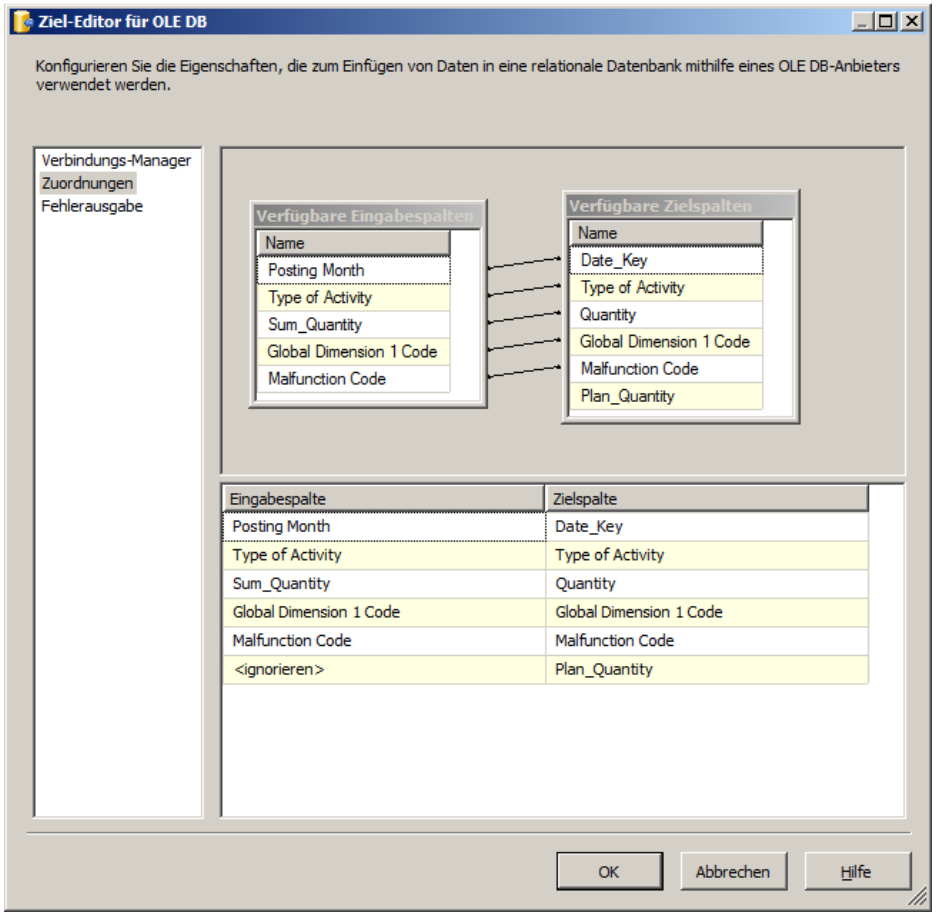

Abbildung 22 OLE-DB Zielspalten Zuordnung

Fazit: Aggregieren über SQL/Aggregationswerkzeug

Das folgende Diagramm soll die Performance der beiden Aggregationsmöglichkeiten darstellen. Die Werte wurden an unterschiedlichen Tagen zu unterschiedlichen Uhrzeiten gemessen und ein Mittelwert bestimmt.

	SSIS	produktiv Datenbank	DWH-Server
CPU	Intel C2D E8400	Intel Xeon X5660	Intel Xeon X5660
Arbeitsspeicher	4GB DDR2	8GB DDR2 ECC	8GB DDR2 ECC
Betriebssystem	Windows Server 2008 R2	Windows Server 2005	Windows Server 2005
DBMS	-	SQL Server 2005	SQL Server 2005
BIDS	Visual Studio 2008	-	-

Tabelle 4 Systemkonfiguration der Testumgebung

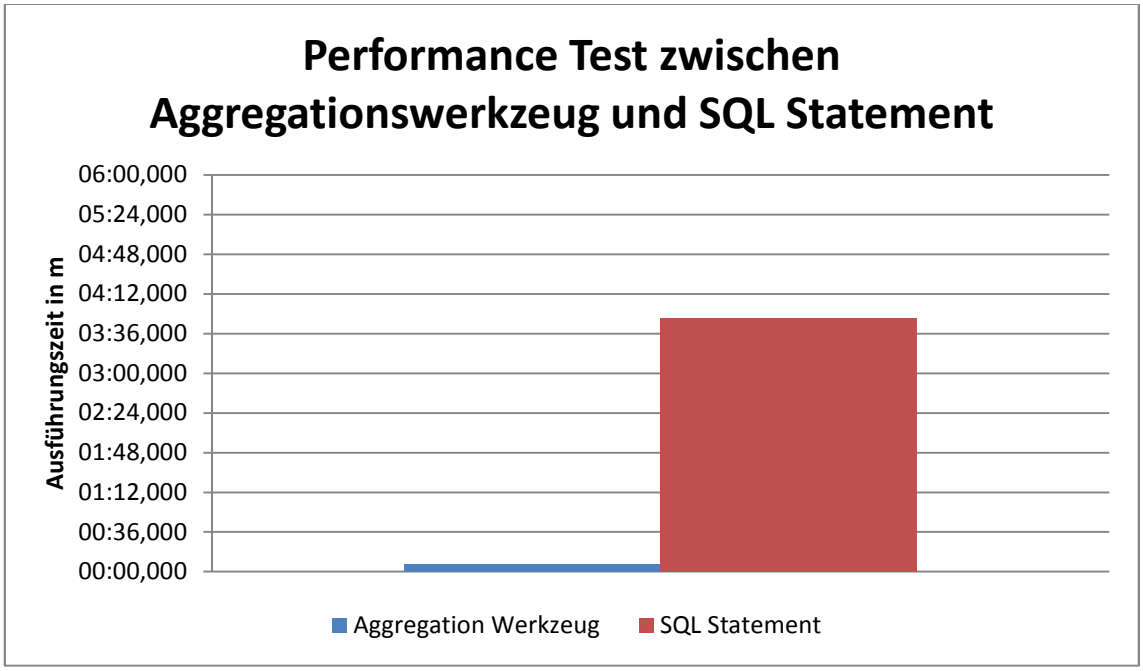

Abbildung 23 Auswertung performance Test

Für diesen Test wurde für die tagesgenaue Aggregierung ein weiterer Datenfluss erstellt, er nimmt die Aggregierung im Extract-Statement der produktiv-Datenbank.

```
1: SELECT [Posting Date], wole.[Type of Activity], SUM(wole.Quantity) AS
   Sum_Quantity, wole.[Global Dimension 1 Code], [Malfunction Code]
2: FROM dbo.[CRONUS TEST SE DD GmbH$Work Order Ledger Entry] AS wole LEFT
   OUTER JOIN
3:     dbo.[CRONUS TEST SE DD GmbH$Work Order Header] AS woh ON wole.[Work Or-
   der No_] = woh.No_
4: GROUP BY wole.[Posting Date], wole.[Type of Activity], wole.[Global Dimen-
   sion 1 Code], [Malfunction Code]
```

Code 4 tagesgenaue Aggregation per SQL Statement

Wie in Abbildung 23 zu erkennen, ist die Verarbeitung auf Seiten des SQL Servers wesentlich langsamer als mit dem SSIS Werkzeug. Die Aufbereitung der Informationen ist also in diesem *speziellen* Fall über die integrierten Werkzeuge des SQL Server Integrated Services zu bevorzugen.

Die Ursache für die lange Laufzeit des SQL Scripts könnte zum Beispiel an den für die Aggregierung schlecht definierten Indizes in der produktiven Datenbank liegen. Bei jedem Projekt sollte geprüft werden, wie sich die Werkzeuge bei anderer Anforderung auf die Laufzeit auswirken.

Planwerte

Einmalige Erzeugung von Beispielwerten

Um für die spätere Auswertung eine Vielzahl von Planwerten vorweisen zu können, werden sie in diesem Prototyp per Zufallsgenerator erstellt. Die Erstellung findet über den SSIS statt.

Ein neuer Datenfluss extrahiert alle vorhandenen Work Order Ledger Entries und gruppiert sie über das Aggregatwerkzeug nach Datum, Leistungsart und Störungscode. Das Aggregatwerkzeug übergibt den nun aggregiert vorliegenden Datenfluss Zeilenweise an das C# Programm. Die Berechnung des Planwertes erfolgt in Abhängigkeit des IST-Wertes, um automatisiert realistisch wirkende Zahlen zu erzeugen. So beträgt der Planwert immer einen Wert zwischen $IST\ Wert * 0.9$ und $IST\ Wert * 1.1$. Nach der Berechnung des Beispielplanwertes für den gegebenen IST-Wert übergibt das C# Programm den berechneten Wert an das OLE DB-Ziel

```
1: public override void Eingabe0_ProcessInputRow(Eingabe0Buffer Row) {
2:     if (Row.AVGQuantity == 0)
3:     {
4:           Row.Quantity = Row.AVGQuantity;
```

```
5:            return;
6:     }
7:     Random random = new Random();
8:     decimal wert = Row.AVGQuantity * (decimal)0.1;
9:     int intWert = Decimal.ToInt32(wert);
10:    int randomNumber = random.Next( (intWert < 0) ? intWert : -(intWert) ,
    (intWert < 0) ? -(intWert) : intWert);
11:    Row.Quantity = (Row.AVGQuantity + randomNumber) * Row.Anzahl;
12: }
```

Code 5 Berechnung Planwerte

Der Planwert ist nur mit den Dimensionen *Zeit*, *Störungscode* und *Leistungsart* verknüpft, die Faktentabellen zusätzlich mit der Kostenstelle (siehe Abbildung 18). Durch diese Gegebenheit teilen sich mehrere Arbeitsauftragspositionen einen Planwert. Aus diesem Grund wird in Code 5, Zeile 11 die Anzahl der Positionen, die sich einen Planwert teilen, mit dem berechneten Planwert für eine Position multipliziert.

Planwerte in der Faktentabelle aktualisieren

Für die Überführung der Planwerte in die Faktentabelle werden die Faktentabellen abgefragt. Durch einen LEFT JOIN der Planwert Tabelle können Planwerte mit den Einträgen der Faktentabelle über das Datum, die Leistungsart und den Störcode verknüpft werden. Um das Beispiel übersichtlicher zu halten, wird von einer Gleichwertigkeit aller Kostenstellen zueinander ausgegangen.

Date_Key	Type of Activity	Malfunction Code	Plan_Quantity	Anzahl
2009-11-01 00:00:00.000	KN-BV		119.000000000000000000000	1
2009-04-01 00:00:00.000	KN-WV	KN F	7.000000000000000000000	1
2010-09-01 00:00:00.000	VERW-ALLG		142453.000000000000000000000	25

Abbildung 24
Beispielhafter Auszug der Quellabfrage für die Aufteilung der Planwerte auf die einzelzelnen Positionen

Damit eine richtige Berechnung der Planwerte, zusätzlich aufgeteilt auf die Kostenstellen, vorgenommen werden kann, muss die Anzahl an Kostenstellen für die Gruppierung [Datum, Leistungsart, Störungscode] bekannt sein.

Das C# Programm wird das Tabellenupdate direkt ausführen und dem Datenfluss nicht die geänderten Daten übergeben. Um eine einheitliche Verbindungsübersicht beizubehalten bietet es sich an, eine ADO.Net Datenverbindung über den Verbindungs-Manager zu erstellen und die Verbindung dem C# Programm zu übergeben.

```
1: public override void Eingabe0_ProcessInputRow(Eingabe0Buffer Row)
2: {
3:     sqlComm = new SqlCommand();
4:     sqlComm.CommandText = "UPDATE DWH_WorkOrderLedger
5:     SET Plan_Quantity = @Quantity WHERE [Date_Key] = @postingDate and [Type
   of Activity] = @ToA and [Malfunction Code] = @Mfc";
6:     sqlComm.Parameters.AddWithValue("@postingDate", Row.DateKey);
7:     sqlComm.Parameters.AddWithValue("@ToA", Row.TypeofActivity);
8:     sqlComm.Parameters.AddWithValue("@Mfc", Row.MalfunctionCode);
9:     sqlComm.Parameters.AddWithValue("@Quantity", (Row.PlanQuantity /
   Row.Anzahl));
10:
11:    sqlComm.Connection = adoNetConnection;
12:    sqlComm.CommandType = CommandType.Text;
13:
14:    sqlComm.ExecuteNonQuery();
15:}
```

Code 6 Aktualisierung der Faktentabellen mit den Planwerten

Die Nutzung der Verbindung und die Ausführung des SQL Statements unterscheiden sich von der Nutzung aus gewöhnlichen C# Programmen nicht.

Dimensionstabellen

Der Ablauf der Übertragung der Dimensionstabellen ist bei den hier benötigten Tabellen sehr ähnlich. Aus diesem Grund wird der Ablauf anhand des Störungscodes näher erläutert.

Bei den ersten Tests ist aufgefallen, dass im MS Dynamics NAV Werte in das Störungscode Feld der Positionen eingetragen wurden, die in der Störungscode-Tabelle nicht existieren. Für die weitere Nutzung der Daten ist eine Fremdschlüsselverbindung der Tabellen zwingend notwendig. Ohne die Verknüpfung können unerwartete Fehler bei der Erstellung von OLAP-Cubes, Berichten oder anderen Auswertungen über den Daten erfolgen. Unter Dynamics NAV sind vermutlich die Fremdschlüssel deaktiviert und so findet keine Prüfung bei Eingabe/Löschung über die Integrität statt. Diese Problematik ist ebenso bei den Kostenstellen und Leistungsarten aufgefallen.

Um das Problem zu umgehen, werden die Störungscodes aus der Störungscode-Tabelle und der Work Order Header Tabelle ausgelesen und miteinander verglichen. Sollten Codes gefunden werden, die in der Störungscode Tabelle nicht existieren, wer-

den über das UNION-Werkzeug beide Datensätze miteinander Verbunden und so in die Zieltabelle übertragen.

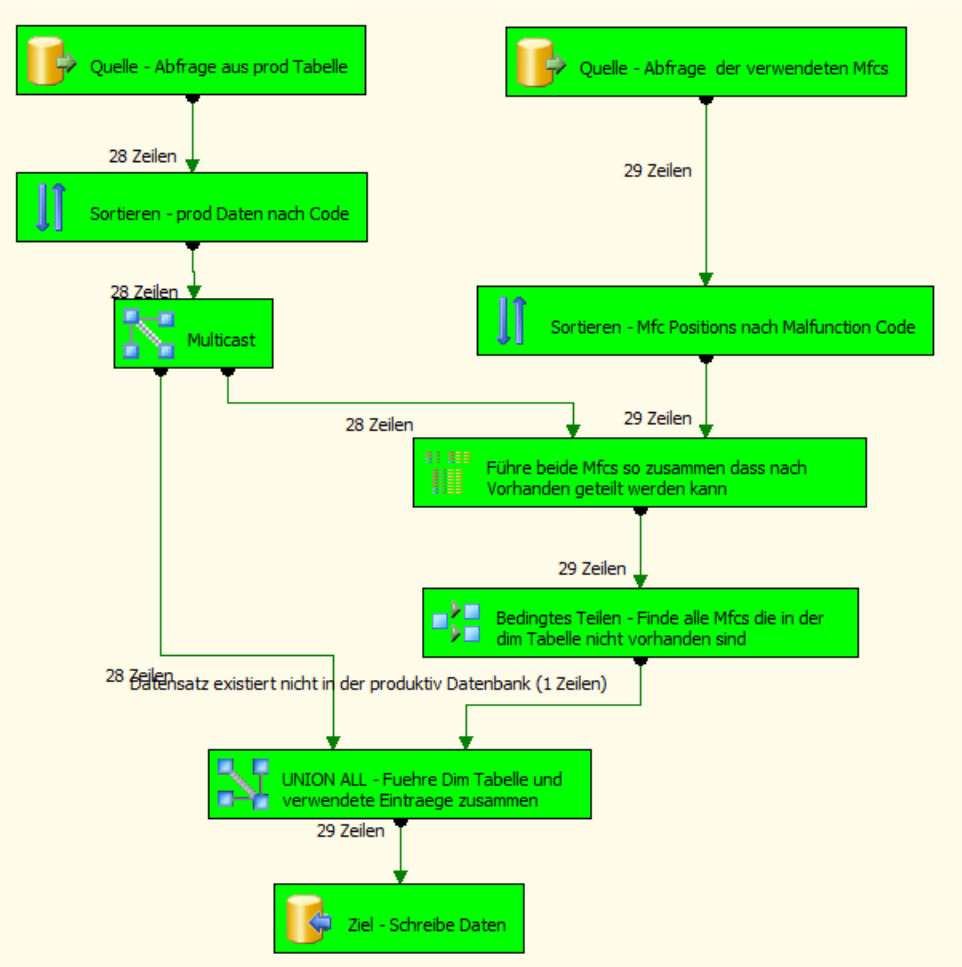

Abbildung 25 Ablauf import Malfunction Code Dimension

Für die Namensgebung der „Description“ Spalte wurde sich entschieden, den Code zu verwenden. Eine fehlerfreie Verknüpfung von Arbeitsauftragsposition und Störungscodes ist nun möglich.

Automatisierte Ausführung des Packages

Ein Data Warehouse muss in regelmäßigen Abständen aktualisiert werden. Die Häufigkeit der Aktualisierung orientiert sich dabei an der Aggregation der Daten und am benötigten Zeitraumes. Werden die Informationen monatsgenau benötigt, muss der ETL-Prozess nicht täglich die Daten aufbereiten. Es genügt einmal am Anfang des Monats, die Daten bis einschl. des Vormonats zu übertragen, genügen.

Das automatisierte Ausführen von Tasks kann über den SQL Server Agent konfiguriert werden. Er wird in der Standardinstallation mit installiert, muss aber unter Umständen noch gestartet werden.

Die Aufträge werden über das Microsoft SQL Server Management Studio verwaltet. Bevor ein neuer Auftrag mit der Ausführung des Packages angelegt werden kann, muss das Package aus dem Visual Studio exportiert und an geeigneter Stelle abgelegt werden. Als Auswahl stehen die Speicherung im Dateisystem und die Speicherung auf einem SQL-Server. Die im Dateisystem gespeicherten Packages lassen sich leichter verwalten, allerdings können Berechtigungen nur auf Dateiebene konfiguriert werden. Bei der zweiten Variante können Berechtigungen auf Paketschutzebene erstellt und Rollen verwendet werden.

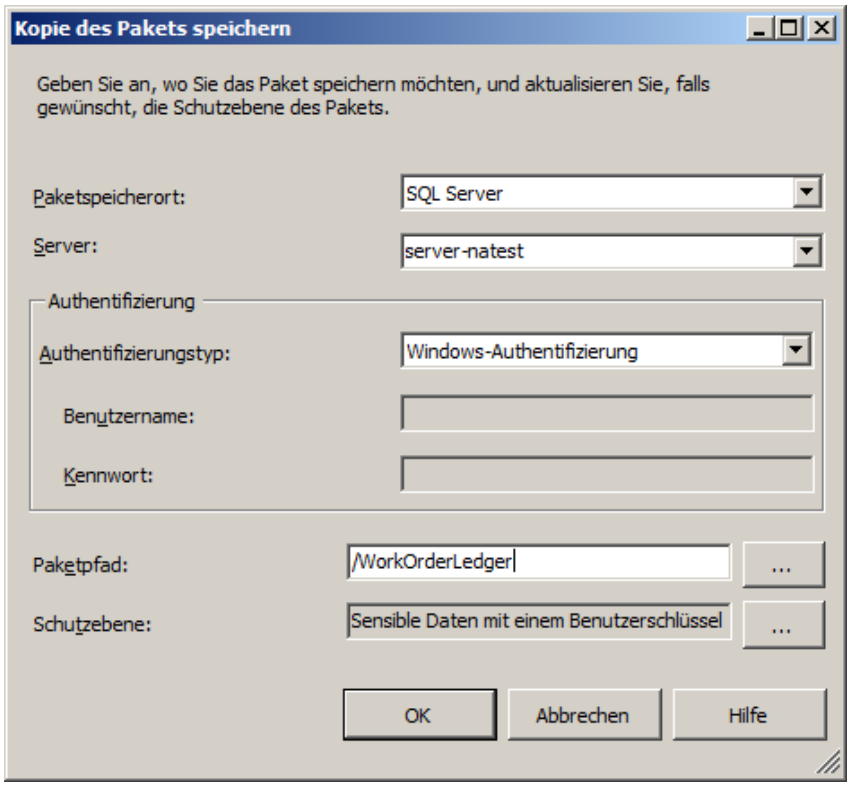

Abbildung 26 dtsx-Paketkopie speichern

Bevor ein Auftrag erstellt wird, muss ein Proxykonto/Anmeldekonto erstellt werden, das die nötigen Rechte besitzt, auf den Ziel-DB Server zu schreiben, den Quell-DB-Server zu lesen und das Package auszuführen. Dazu wird ein Domäne-Konto über „Sicherheit -> Anmeldeinformationen“ hinzugefügt. Im Anschluss wird über „SQL Server-Agent -> Proxys“ ein neues Proxykonto eingerichtet.

Der Auftrag wird über „Aufträge -> Neu“ angelegt. Es muss das Feld „Name“ ausgefüllt sein. Die auszuführenden Packages werden über „Schritte“ hinzugefügt. Die Schritte

werden sequenziell ausgeführt. Zu beachten ist, dass der Proxyaccount unter „Ausführen als“ eingetragen ist. Um eine regelmäßige Ausführung zu gewährleisten, wird ein Auftragszeitplan erstellt. Der Zeitpunkt der Ausführung ist der 3. eines jeden Monats.

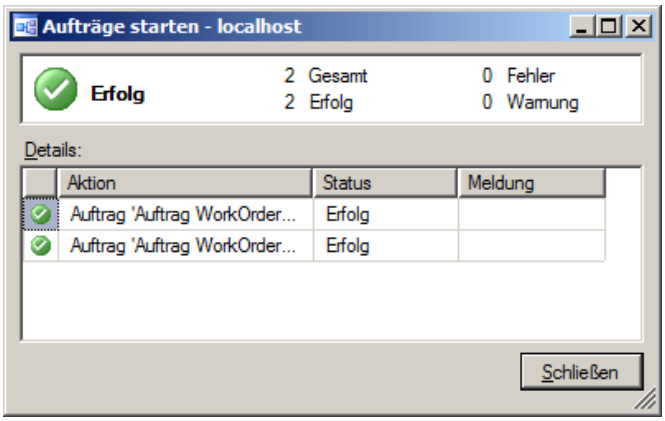

Abbildung 27 Manuelle Ausführung eines Auftrages

Zur Prüfung, ob alle Parameter richtig gesetzt wurden, sollte der Auftrag einmalig manuell ausgeführt werden.

Abschlussbemerkung

Es befinden sich jetzt alle Daten in aggregierter Form im Data Warehouse. Auf das Data Warehouse kann mit unterschiedlichen Analyse- und Reporting Tools diverser Hersteller zugegriffen werden. Eine Bindung der Auswertungstools aus dem Microsoft Portfolio ist nicht gegeben. Auf die Datenbank kann mittels SQL Server Management Studio zugegriffen werden. Das Exportieren der Daten für eine Migration in ein anderes DBMS ist ebenfalls ohne Zusatzkosten möglich.

5.3 OLAP-Cube Entwurf

In einem OLAP-Cube werden die Informationen multidimensional gespeichert. Die Zugriffszeit ist sehr gering, und es wird ein „Navigieren“ durch die Daten ermöglicht. Wie auch die SSIS Entwicklung werden Cubes im Visual Studio erstellt. Hierfür wird ein neues Analysis Services-Projekt angelegt.

5.3.1 Datenquellen und Datenquellensichten

Im ersten Schritt muss eine Datenquelle definiert werden. Für diesen Prototype wird eine Verbindung zur Data Warehouse Datenbank konfiguriert.

Damit der spätere Cube auf die Quelldaten zugreifen kann, muss eine Datenquellensicht erstellt werden. In sie werden alle benötigten Tabellen hinzugefügt. Sollten die Tabellen in der Quelldatenbank in Beziehung zu einander stehen, wird sie auch in der Datenquellansicht angezeigt und erstellt.

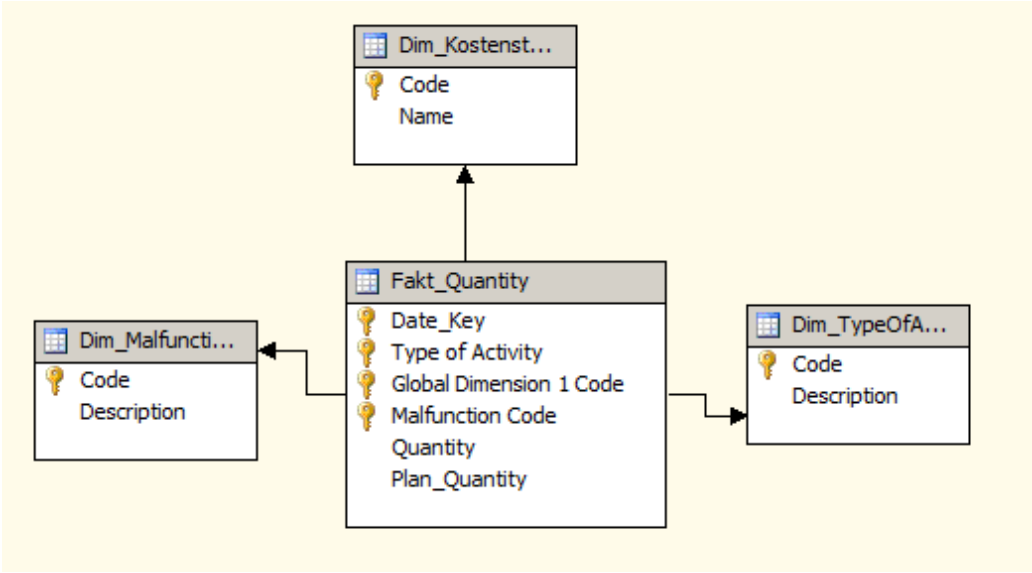

Abbildung 28 Sternschema der Quelldaten in der Datenquellansicht

Die Verknüpfungen finden üblicherweise über die Primär/Fremdschlüssel statt.

5.3.2 Dimensionen

Durch Dimensionen kann das Sichtfeld und die Perspektive auf die Daten eingeschränkt bzw. gefiltert werden. Dimensionen können neben einer flachen Struktur auch hierarchisch aufgebaut sein. Bei dem Erstellen eines Cubes in Visual Studio werden

die von VS gefundenen Dimensionen automatisch aus den Quelltabellen angelegt. Sollten Dimensionen von Visual Studio nicht automatisch als solche erkannt werden können sie angelegt werden.

Als Besonderheit gelten Zeiträume. Eine Kalenderdimension muss manuell erstellt werden, sofern keine im Data Warehouse existiert. Ohne eine Kalenderdimension ist es nicht möglich, eine Zeit-Hierarchie, wie z.B. Jahr – Monat – Tag, aufzubauen. Microsoft bietet für das Erstellen der Kalender Dimension ein Werkzeug an. Über den Dimensions-Assistenten kann eine Zeittabelle in der Quelldatenbank oder direkt auf den Analysis Services angelegt werden (sofern Rechte vorhanden).

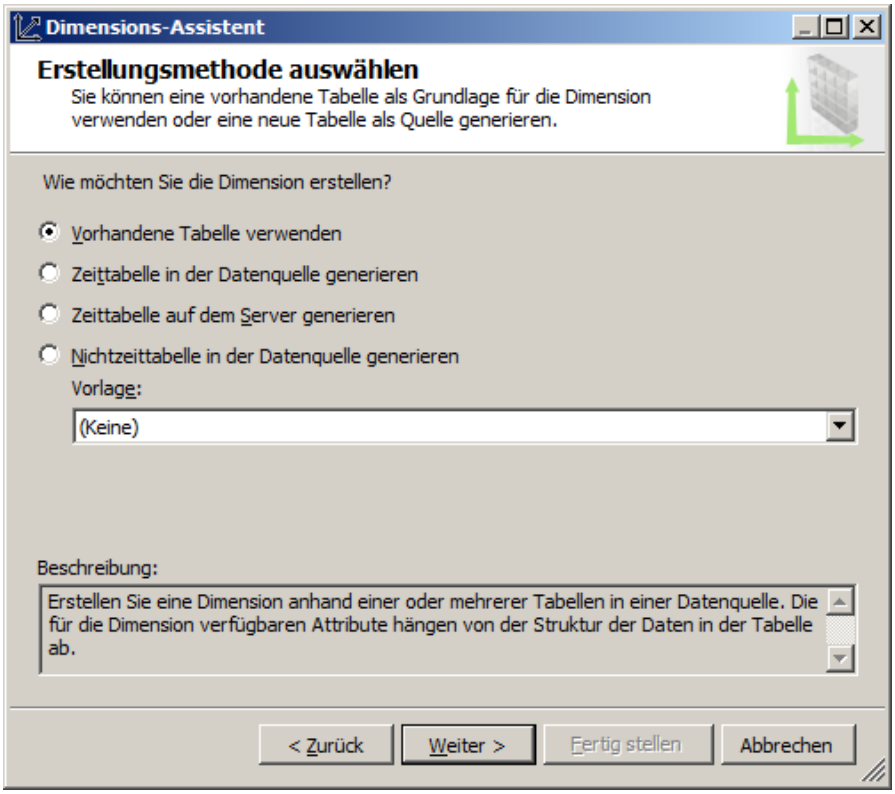

Abbildung 29 Dimension anlegen - Zeittabelle

Während der Erstellung der Zeitdimension müssen der erste und letzte Kalendertag, sowie die abzubildenden Zeiträume definiert werden. Das Element „Datum“ ist hierbei ein Pflichtfeld, das benötigt wird, um die Zeittabelle mit der Faktentabelle zu verknüpfen. Wenn eine tagesgenaue Auswertung nicht nötig oder möglich ist, kann das Feld in einem späteren Schritt unsichtbar gestellt werden. Für den Prototyp wurden die Zeiträume „Jahr – Quartal – Monat“ gewählt, da die Daten im Datawarehouse monatsgenau vorliegen.

Damit der Monat die tiefste Hierarchie-Ebene darstellt und somit die Auswahl begrenzt wird, muss die Hierarchie der Dimension angepasst werden.

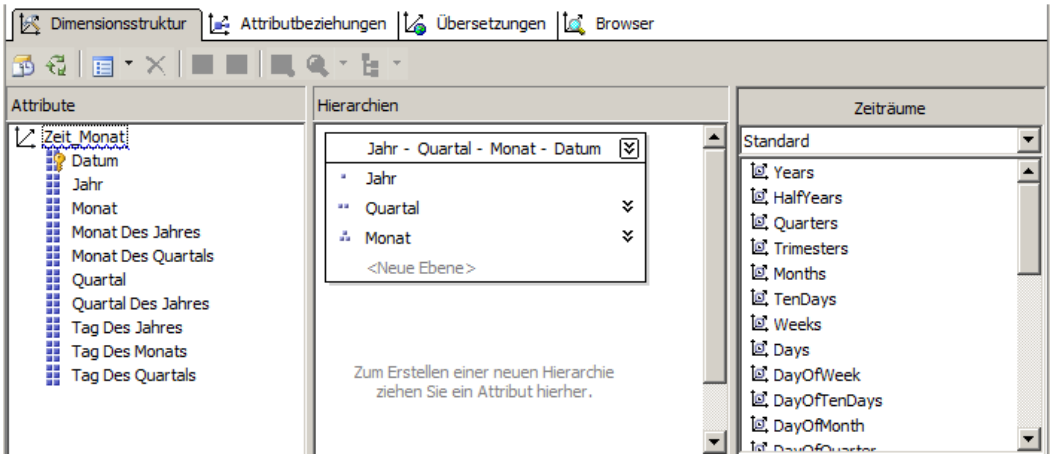

Abbildung 30 Dimensionsstrukur der Zeitdimension

Die restlichen Dimensionen des Prototyps werden automatisch bei der Erstellung des Cubes angelegt.

5.3.3 Cube

Der Cube kann mittels Cube-Assistenten erstellt werden. Er wird aus den vorhandenen Tabellen erstellt. Als Measuretabelle wählt man die Faktentabelle ‚Fakt_Quantity', die Dimensionen werden aus den Tabellen ‚*Dim_MalfunctionCode*', ‚*Dim_Kostenstellen*' und ‚*Dim_TypeOfActivity*' gebildet. Nachdem der Cube erstellt wurde, erhält man in dem Tab „CubeStruktur" einen ersten Überblick über die Dimensionen und Fakten. In dieser Übersicht können weitere Dimensionen und Measures hinzugefügt werden. Auf diesem Weg wird die Zeit-Dimension dem Cube hinzugefügt.

Im Tab „Dimensionsverwendung" werden die Verknüpfungen definiert. Die mit dem Assistenten erstellten Dimensionen sollten anhand der Verknüpfungen in der Datenquellensicht bereits korrekt verknüpft sein. Die erst später hinzugefügte Zeit-Dimension muss hier noch korrekt mit den Fakten verknüpft werden. Als Verknüpfung wählt man einen regulären Beziehungstyp mit den Spalten Datum (Zeit-Dimension) und Date_Key (Faktentabelle).

Um den Cube zu testen, muss er mittels [F5]-Taste aufgebaut werden. Danach kann er über den integrierten Cubebrowser des Visual Studios getestet werden. Der Browser stellt einen pivot-Table Editor bereit. Mittels Drag-and-Drop können Dimensionen und Fakten der Ansicht hinzugefügt werden. Er dient in erster Linie dem Entwickler des Cubes zu Testzwecken und nicht als Auswertungstool des Endanwenders.

Leistungsart	Störungscode	Kostenstelle
(Mehrere Elemente)	All	All

Spaltenfelder hierher ziehen

Jahr	Quartal	Monat	Quantity	Plan Quantity
⊞ Kalender 2007			2630,95	2619
⊞ Kalender 2010			59	57
⊟ Kalender 2011	⊟ Quartal 3, 2011	September 2011	10896,5	10177
		Gesamt	10896,5	10177
	⊟ Quartal 4, 2011	Oktober 2011	10878,63	10940
		November 2011	16424,97	16804
		Gesamt	27303,6	27744
	Gesamt		38200,1	37921
Gesamtergebnis			40890,05	40597

Abbildung 31 Cube Browser des Visual Studios

In vielen Fällen werden weitere berechenbare Werte benötigt. Interessante Kennzahlen sind zum Beispiel der Vormonatsverbrauch, oder der Verbrauch des gleichen Zeitraumes im letzten Jahr. Diese Kennzahlen können in absoluten Zahlen ausgegeben oder prozentual berechnet werden. Für diese Fälle können dem Cube „berechnete Elemente“ hinzugefügt werden. Ein berechnetes Element stellt eine neue Kennzahl zur Verfügung. Die Berechnung oder Filterung der auszugebenen Kennzahlen erfolgt mittels MDX. MDX steht für Multidimensional Expressions und ist die Datenbanksprache für OLAP.

Um in diesem Beispiel die benötigten Stunden des Vorjahreszeitraumes zu berechnen, wird ein neues berechnetes Element mit dem Namen „Quantity Last YTP“ – YTP steht hierbei für Year Time Period.

Der folgende Ausdruck berechnet den Verbrauch des Vorjahreszeitraumes:

```
1: (
2:      [Measures].[Quantity],
3:     ParallelPeriod([Zeit_Monat].[Jahr - Quartal - Monat - Datum].[Jahr])
4: )
```

Code 7 Berechnung Vorjahreszeitraum

Die Berechnung der prozentualen Steigerung zum Vorjahr erfolgt ebenso über ein berechnetes Element. In diesem Element wird Bezug auf die „Quantity Last YTP“ genommen und mittels der Formel

$$\frac{Vorjahreszeitraum - gewählter\ Zeitraum}{Vorjahreszeitraum}$$

berechnet. Als Formatzeichenfolge wird „Percent“ gewählt, was zur Folge hat, dass die berechneten Werte mit 100 multipliziert werden und ein %-Zeichen angehangen wird.

```
1: (
2:    ([Measures].[Plan Quantity]-[Measures].[Quantity Last
   YTP])/[Measures].[Quantity Last YTP]
3: )
```

Code 8 Berechnung prozenduale Steigerung

Nachdem der Cube neu erstellt wurde, kann die neuangelegte Kennzahl zur Überprüfung der Pivot Tabelle hinzugefügt werden. Die angelegten Kennzahlen können wie feste Kennzahlen aus den Tabellen genutzt werden.

Kostenstelle	Störungscode	Leistungsart
All	(Mehrere Elemente)	All

Spaltenfelder hierher ziehen

Jahr	Quartal	Monat	Quantity	Quantity Last YTP	Quantity Percent Last YTP
⊞ Kalender 2007			15631,84	978,65	1439,88%
⊞ Kalender 2008			76838,69	15631,84	385,67%
⊟ Kalender 2009	⊞ Quartal 1, 2009		5263,31	46320,03	-88,68%
	⊟ Quartal 2, 2009	April 2009	1685,45	504,5	235,58%
		Mai 2009	1113,75	161	561,49%
		Juni 2009	2416,52	549,75	321,65%
		Gesamt	5215,72	1215,25	317,69%
	⊞ Quartal 3, 2009		25661,17	13479,83	87,53%
	⊞ Quartal 4, 2009		7693,26	15823,58	-54,42%
	Gesamt		43833,46	76838,69	-44,28%
⊟ Kalender 2010	⊞ Quartal 1, 2010		4646,16	5263,31	-14,84%
	⊟ Quartal 2, 2010	April 2010	609,85	1685,45	-65,53%
		Mai 2010	8925,55	1113,75	648,37%
		Juni 2010	8953,48	2416,52	249,68%
		Gesamt	18488,88	5215,72	232,95%
	⊞ Quartal 3, 2010		12468,77	25661,17	-51,61%
	⊞ Quartal 4, 2010		12090,78	7693,26	58,88%
	Gesamt		47694,59	43833,46	6,06%
⊞ Kalender 2011			111596,38	47694,59	127,12%
⊞ Kalender 2012			20842,05	111596,38	-82,30%
Gesamtergebnis			316437,01		1,#INF

Abbildung 32 Quantity / Quantity Last YTP / Quantity Percent Last YTP

5.3.4 Bereitstellung

Bevor der Cube von Frontendwerkzeugen genutzt werden kann, muss er veröffentlicht werden. Die Veröffentlichung erfolgt in zwei bzw. in drei Schritten.

Analysis Projekt im BIDS erstellen

Bevor ein Analysis Projekt auf den Analysis Services veröffentlicht werden kann, muss es aus der Entwicklungsumgebung heraus erstellt werden. Die Erstellung erfolgt über das Menü „Erstellen“.

Nachdem das Projekt erfolgreich erstellt wurde, befinden sich alle notwendigen Dateien im *bin*-Verzeichnis des Projektes.

Die Datenbankdefinitionen befinden sich in der *.asdatabase-Datei, zusätzliche Projektkonfigurationen befinden sich in den weiteren Dateien.

Analysis Projekt in den Analysis Services bereitstellen

Die Bereitstellung des Analysis Projekts in den Analysis Services geschieht über den Deployment-Wizard. Während der Bereitstellungen werden die Datenbankdefinitionsdatei und die zugehörigen Definitionsdateien eingelesen und ausgewertet. Anschließend kann ein Identitätswechsel für die Quelldatenbanken eingerichtet und die Zieldatenbank ausgewählt werden.

Periodische Aktualisierung der Cube Daten

Der erfolgreich erstellte Cube enthält alle Daten die zum Zeitpunkt der Erstellung im Data Warehouse vorhanden waren. Bei Änderungen der Daten wird die Analysis Services Datenbank nicht unterrichtet. Damit die Aktualisierung des Cubes nicht bei jeder Data Warehouse Änderung manuell ausgeführt werden muss, kann ein „Proactive Caching“ für einzelne Faktenpartitionen hinzugefügt werden.

Da für diesen Anwendungsfall die Daten im Data Warehouse nur einmal aller 30 Tage aufbereitet werden, muss auch der OLAP-Cube nur einmal im Monat aktualisiert werden. Alternativ kann für andere Anwendungsfälle die Intervallszeit auch auf wenige Sekunden gesetzt werden.

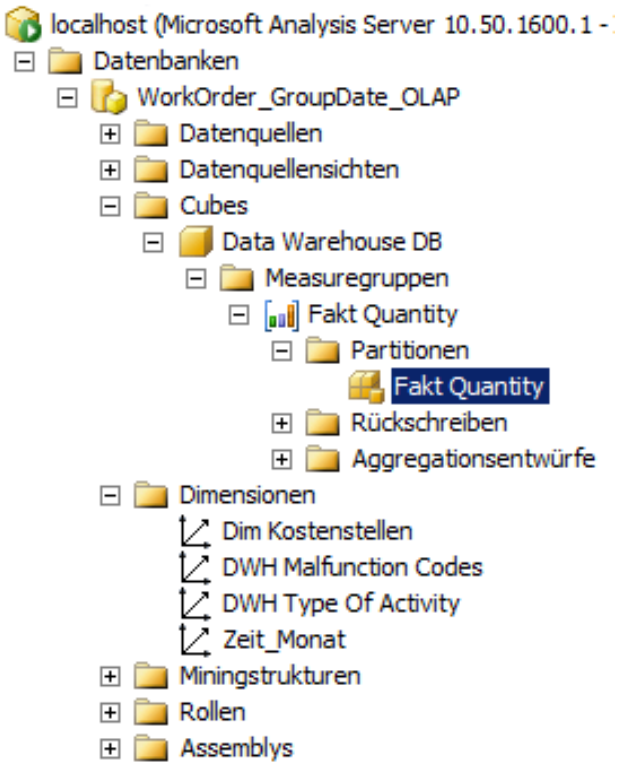

Abbildung 33 Analysis Projekt Struktur im Analysis Services

5.4 Berichtsentwurf

Die Entwicklung des Berichts findet im Business Intelligence Development Studio statt. Für die grundlegende Erstellung wird der Berichts-Assistent verwendet. Der Assistent stellt einen grafischen Abfragegenerator zur Verfügung. Nachdem die Verbindung zur Analysis Services Datenbank hergestellt wurde, werden mittels Drag-and-Drop die Kennzahlen und Dimensionen dem Bericht hinzugefügt.

5.4.1 Berichtsgenerierung

Um eine spätere Filterung durch den Nutzer zuzulassen, müssen die Dimensionen Zeitraum mit dem Element Jahr, die Leistungsart, der Störungscode und die Kostenstelle als Parameter hinzugefügt werden.

Nachdem der Generator den Grundaufbau des Berichts erstellt hat, muss er an die eigenen Bedürfnisse angepasst werden. Hierfür wird eine Kopfzeile mit dem Logo der Stadtentwässerung Dresden GmbH und dem Namen des Berichts hinzugefügt.

Für die Informationen über die Ausführungszeit und den angemeldeten Nutzer wird eine Fußzeile angelegt. Die benötigten Informationen werden in globalen Berichtsvari-

ablen vorgehalten. Alle vorhandenen Variablen können im Ausdrucksеditor unter „Integrierte Felder“ nachgesehen werden.

Der Bericht kann durch die vier Parameter von dem Nutzer individualisiert werden. Damit eine spätere Zuordnung zwischen den gewählten Parametern und den ausgegebenen Kennwerten möglich ist, müssen die gewählten Parameter auf dem Bericht notiert sein. Der Zugriff auf die gewählten Parameter geschieht über den Namespace „Parameters“. Alle dem Bericht hinzugefügten Parameter können so angesprochen werden.

Für jeden Parameter muss unter der Tabelle ein Textfeld angelegt und mit einem Ausdruck belegt werden.

```
1: =Join(Parameters!ZeitMonatJahr.Label, ",")
```

Code 9 Ausdruck: Ausgewählte Jahre

Parameter mit Mehrfachauswahl werden wie Arrays behandelt. Der Join-Befehl aus Code 9 verknüpft die Array-Elemente mit einem Komma zu einem String.

5.4.2 Formatierung

Die vom Assistenten erstellte Tabelle wird für eine bessere Lesbarkeit formatiert. In jedem Tabellenfeld können statische Werte, Ausdrücke oder Feldbeziehungen hinterlegt werden.

Zeitspalten

In der Analysis Services Datenbank wurden zu viele Informationen für jedes Element der Zeitdimension in der Bezeichnungsspalte hinterlegt. So steht zum Beispiel für das Jahr 2010 die Bezeichnung „Kalender 2010“ und für das 4. Quartal 2010 die Bezeichnung „Quartal 4, 2010“. Der Bericht bildet die Zeitdimension in einer Hierarchie ab. Deshalb ist die Jahreszahl hinter der Quartalszahl unnötig. Aufgrund der Tabellenstruktur ist sofort das zugehörige Jahr erkennbar (siehe Abbildung 34 Zeithierarchie Berichtsansicht).

Die überflüssigen Informationen werden mittels Text-Funktionen entfernt.

```
1: =Left(Fields!Quartal.Value, 9)
```

Code 10 Entfernung der Jahreszahl hinter der Quartalsbezeichnung

Die in Code 10 verwendete Left-Funktion entfernt alle Zeichen aus der übergebenen Zeichenkette, die nach dem 9. Zeichen von links stehen. Ähnliche Ausdrücke werden für die Formatierung des Jahres und des Monats verwendet.

Jahr	Quartal	Monat	Menge	Plan Menge	Menge l. Jahr Periode	Menge % l. Jahr Periode
2010			85830405,97	85040356	97589425,19	-55,1%
	Quartal 1		15051791,55	14680878	14794781,26	1,97%
		Januar	3555626,74	3485903	3173291,37	9,85%
		Februar	4241509,73	4064767	4277636,12	-4,98%
		März	7254655,08	7130208	7343853,77	-2,91%

Abbildung 34 Zeithierarchie Berichtsansicht

Mengenspalten

Die in den Quelldaten vorliegenden Mengen haben sehr viele Nachkommastellen. Benötigt wird für diesen Bericht eine Genauigkeit von zwei Nachkommastellen. Nachkommastellen können mit der Zahl-Funktion „Round“ eliminiert werden. Durch Angabe der Kommastellen als Parameter rundet die Funktion auf die gewünschte Anzahl.

Um die Übersichtlichkeit weiter zu erhöhen, sollen die summierten Mengen der Quartale und Jahre berechnet werden (siehe Abbildung 34, 2010 und Quartal 1 Zeile).

```
1: =Round(Sum(Fields!Quantity.Value),2)
```

Code 11 Menge summieren

Der in Code 11 verwendete Ausdruck summiert alle Mengen der ihm untergeordneten Hierarchiestufe und rundet das Ergebnis auf zwei Nachkommastellen.

Dieser Ausdruck, mit abgewandelter Field-Verknüpfung, wird für jede der vier Spalten Menge, Plan Menge, Menge l. Jahr Periode und Menge % l. Jahr Periode verwendet.

Entwurfsansicht

Der Bericht wurde über die Entwurfsansicht des Business Intelligence Development Studios entwickelt. Es befinden sich ein Seitenkopf und ein Seitenfuß mit nützlichen allgemeinen Informationen, eine Datentabelle mit den Kennwerten und eine Übersicht über die ausgewählten Parameter in dem Entwurf.

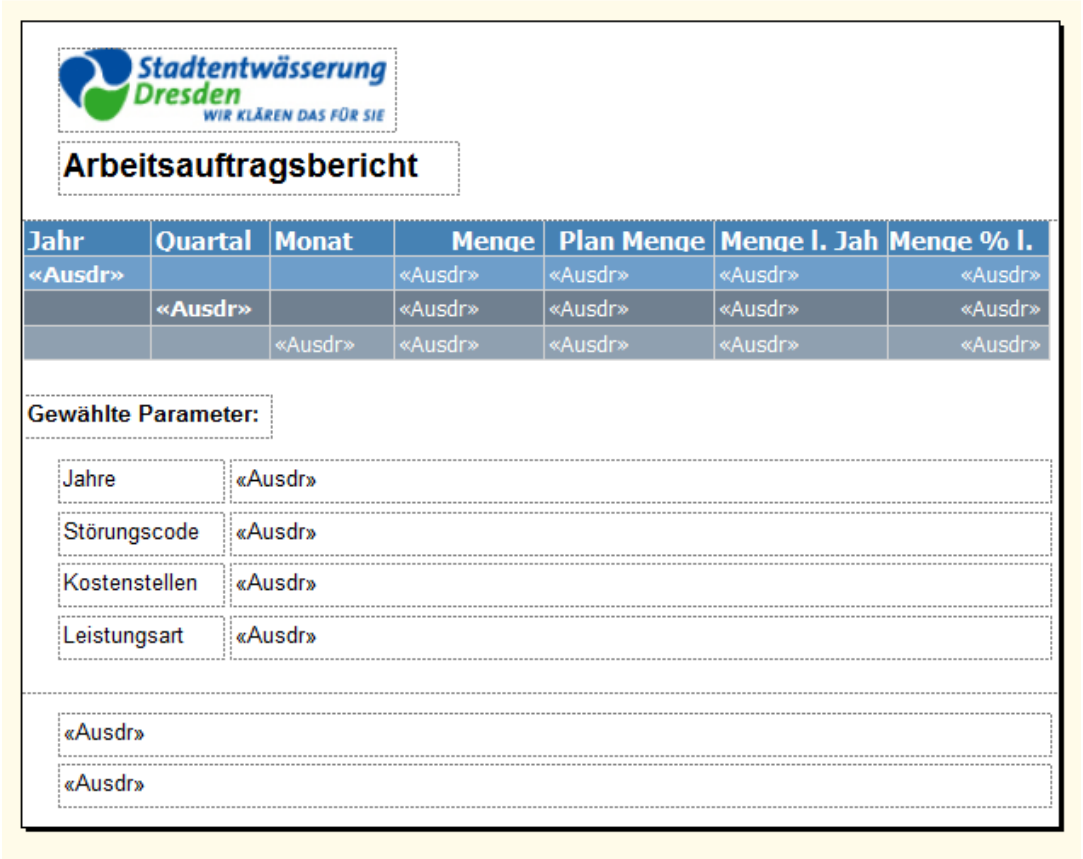

Abbildung 35 Entwurfsansicht

Für eine Überprüfung wird der Bericht über die Option „Vorschau“ generiert. In dieser Vorschau wird der Bericht so angezeigt, wie er später auf den Reporting Services erscheinen wird.

6 Fazit

Bei der Stadtentwässerung Dresden GmbH wird eine Vielzahl von Berichten monatlich bzw. jährlich benötigt. Zurzeit werden alle vorhandenen Berichte über den Reporting Service der Microsoft SQL Server abgewickelt. Auf die Verwendung eines Data Warehouse oder einer zusätzlichen OLAP Datenhaltung wird verzichtet.

Viele der benötigten Daten befinden sich in MS Dynamics NAV. Weitere Daten werden über Excel Tabellen oder von den Sachbearbeitern manuell eingepflegt. Die Datenbankstruktur von MS Dynamics NAV ist für eine Berichtserstellung ungeeignet (siehe Abbildung 36).

Um die Stärken der OLAP-Abfragen aufzuzeigen, wurde ein Testprogramm in C# entwickelt (siehe Anlage 1). Die Zeitmessungen für Abbildung 36 wurden mit den Abfragen aus Anlage 2 gemessen. Hierbei ist zu beachten, dass die Abfragen ohne Parameter ausgeführt wurden. Außerdem wurde auf die Auswertung der „Last Year Time Periode“ verzichtet.

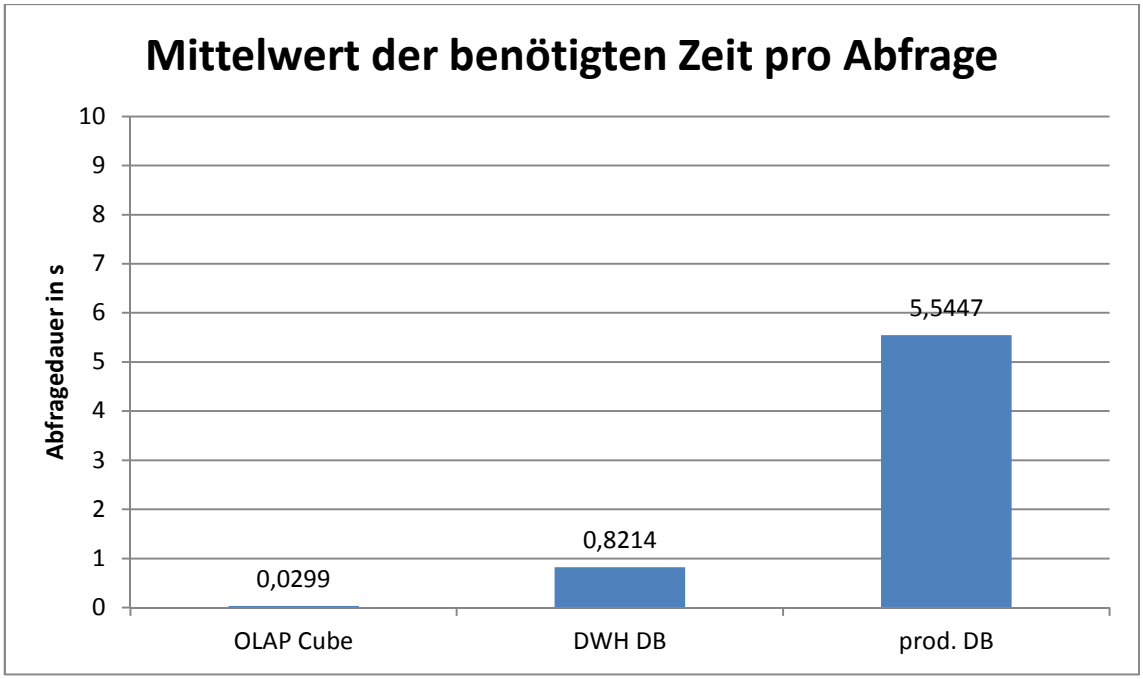

Abbildung 36 Abfragezeit Vergleich zwischen OLAP, Data Warehouse und produktiv Datenbank

Mit den vernachlässigten Werten würde die Abfrage aus den produktiven NAV Daten weiter an Komplexität gewinnen und die Fehlerwahrscheinlichkeit steigen.

Für eine effiziente Auswertungsleistung ist die Überführung der Daten in ein geeigneteres Format unabdingbar.

Literaturverzeichnis

"Michael". *Stackoverflow*. 13. Dezember 2011. 28. August 2012. <http://stackoverflow.com/questions/8483402/ssrs-showing-the-correct-execution-time-on-a-two-page-report>.

Alexander, Sascha. „Checkliste für Business Intelligence.“ *Tecchannel Compact 01/2010* Januar 2010: 65-67.

Azevedo, Pedro und Berhard Brosius. *Business Intelligence und Reporting mit SQL Server 2008*. Unterschleißheim: Microsoft Press Deutschland, 2009.

Azevedo, Pedro, et al. *Business Intelligence und Reporting mit Microsoft SQL Server 2005*. Unterschleißheim: Microsoft Press, 2006.

Born, Achim und Jürgen Diercks. „Business Intelligence, Analyse und Reporting: Stand der Dinge.“ *iX Magazin für professionelle Informationstechnik* 01/2012 (2012): 90-96.

Chamoni, Peter und Peter Gluchowski. *Analytische Informationssysteme - Business Intelligence Technologien und Anwendungen*. Duisburg: Springer, 2006.

Codd, E.F., S.B. Codd und C.T. Salley. *Providing OLAP to User-Analysts: An IT Mandate*. 1993. 11. Juli 2012. <http://www.minet.uni-jena.de/dbis/lehre/ss2005/sem_dwh/lit/Cod93.pdf>.

Corporate Planning. „Modulblatt CP-BSC.“ 30. 07 2012. <http://www.corporate-planning.com/download/software/modulblaetter/CP-BSC.pdf>.

hyperspace. *Balanced Scorecard mit hyScore*. 30. 07 2012. <http://www.hyperspace.de/de/balanced_scorecard.htm>.

Inmon, William H. *Building the Data Warehouse*. 3. New York, Chichester: Wiley & Sons, 2005.

Kemper, Hans-Georg, Henning Baars und Walid Mehanna. *Business Intelligence - Grundlagen und praktische Anwendungen*. 3. Wiesbaden: Vieweg+Teubner Verlag; Springer Fachmedien Wiesbaden GmbH, 2010.

Manhart, Klaus. *Tecchannel OLAP-Frontends*. 16. April 2008. 03. September 2012. <http://www.tecchannel.de/server/sql/1751285/bi_methoden_teil_1_ad_hoc_analysen_mit_olap/index12.html>.

Manhart, Klaus und Mark Zimmermann. *Basiswissen SOA - BI - CRM - ECM : Grundlagen, Methoden, Praxis; Tec Channel*. München: Tredition-Verl.; IDG Business Media, 2009.

Thurnheer von Berneck, Andreas. „Temporale Auswertungsformen in OLAP.“ 2003. *edoc.unibas.ch.* PDF. 19. Juli 2012. <http://edoc.unibas.ch/54/1/DissB_6469.pdf>.

Anlagen

1 C# Quelltext Performance Test

```
1: using System;
2: using System.Collections.Generic;
3: using System.Linq;
4: using System.Text;
5: using System.Configuration;
6: using System.Data.SqlClient;
7: using Microsoft.AnalysisServices.AdomdClient;
8:
9: namespace Performance_Messprogramm
10:{
11:    class Program
12:    {
13:        static void Main(string[] args)
14:        {
15:            int intTimes = 100;
16:            Console.Clear();
17:            Program prgm = new Program();
18:            prgm.Start(intTimes);
19:        }
20:
21:        public void Start(int intTimes)
22:        {
23:            AppSettingsReader config = new AppSettingsReader();
24:
25:            // Initialize Databases
26:            MsSql sqlLog = new MsSql(config.GetValue("SQLServer_Log",
   typeof(string)).ToString(),
27:                                     config.GetValue("SQLDatabase_Log",
   typeof(string)).ToString(),
28:                                     con-
   fig.GetValue("SQLServerInstance_Log", typeof(string)).ToString());
29:
30:            MsSql sqlDwh = new MsSql(config.GetValue("SQLServer_DWH",
   typeof(string)).ToString(),
31:                                     config.GetValue("SQLDatabase_DWH",
   typeof(string)).ToString(),
```

```
32:                                     con-
   fig.GetValue("SQLServerInstance_DWH", typeof(string)).ToString());
33:
34:             MsSql sqlProd = new MsSql(config.GetValue("SQLServer_Prod",
   typeof(string)).ToString(),
35:                                     config.GetValue("SQLDatabase_Prod",
   typeof(string)).ToString(),
36:                                     con-
   fig.GetValue("SQLServerInstance_Prod", typeof(string)).ToString());
37:
38:             AdomdConnection adoCon = new AdomdConnection("Data Source=" +
   config.GetValue("SQLServer_OLAP", typeof(string)).ToString()+
39:
   ";Provider=MSOLAP;Catalog=" + config.GetValue("SQLDatabase_OLAP",
   typeof(string)).ToString()+
40:                                                             ";Cube="+ con-
   fig.GetValue("SQLCube_OLAP", typeof(string)).ToString());
41:             adoCon.Open();
42:
43:             // Variables
44:             DateTime dtStart;
45:             DateTime dtEnd;
46:             double dDtResult;
47:
48:             Console.WriteLine("Erstelle Messtabelle ... ");
49:             sqlLog.NonQuery("DELETE FROM DWH_PerformanceTest");
50:
51:             Console.WriteLine("Starte Test");
52:
53:             SqlDataReader dr;
54:
55:             for (int i = 0; i < intTimes; i++)
56:             {
57:                 Console.WriteLine("Run: " + (i + 1));
58:
59:                 //// DWH Abfrage
60:                 // Cache Leeren;
61:                 sqlDwh.NonQuery("dbcc freeproccache");
62:                 sqlDwh.NonQuery("dbcc dropcleanbuffers");
63:                 System.Threading.Thread.Sleep(500);
64:                 dtStart = DateTime.Now;
65:                 //
```

```
66:                 #region sqlAbfrage
67:                 dr = sqlDwh.Query(_DB_ABFRAGE_1);
68:                 #endregion
69:                 dtEnd = DateTime.Now;
70:                 dDtResult = calcDurationTime(dtStart, dtEnd);
71:                 saveResult(sqlLog, "DWH", dDtResult);
72:                 //// DWH Abfrage Ende
73:
74:                 //// OLAP Abfrage
75:
76:                 // clr cache
77:                 AdomdCommand adoCCCmd = adoCon.CreateCommand();
78:                 adoCCCmd.CommandText = "<ClearCache
    xmlns=\"http://schemas.microsoft.com/analysisservices/2003/engine\">" + @"
79:                                                 <Object>
80:                                                 <Data-
    baseID>"+config.GetValue("SQLDatabase_OLAP",
    typeof(string)).ToString()+@"</DatabaseID>
81:                                                 </Object>
82:                                             </ClearCache>";
83:                 adoCCCmd.ExecuteNonQuery();
84:                 System.Threading.Thread.Sleep(500);
85:                 // -->
86:                 AdomdCommand adoCmd = adoCon.CreateCommand();
87:                 adoCmd.CommandText = _DB_ABFRAGE_2;
88:                 dtStart = DateTime.Now;
89:                 adoCmd.ExecuteNonQuery();
90:                 dtEnd = DateTime.Now;
91:                 dDtResult = calcDurationTime(dtStart, dtEnd);
92:                 saveResult(sqlLog, "OLAP", dDtResult);
93:                 // OLAP Abfrage Ende
94:
95:                 // prod. DB Abfrage
96:                 sqlProd.NonQuery("dbcc freeproccache");
97:                 sqlProd.NonQuery("dbcc dropcleanbuffers");
98:                 System.Threading.Thread.Sleep(500);
99:                 dtStart = DateTime.Now;
100:                    //
101:                    #region sqlAbfrage
102:                    dr = sqlProd.Query(_DB_ABFRAGE_3);
103:                    #endregion
```

```
104:                    dtEnd = DateTime.Now;
105:                    dDtResult = calcDurationTime(dtStart, dtEnd);
106:                    saveResult(sqlLog, "PROD", dDtResult);
107:                    //
108:                }
109:                Console.Read();
110:            }
111:
112:            private double calcDurationTime(DateTime dtStart, DateTime
    dtEnd)
113:            {
114:
115:                return dtEnd.Subtract(dtStart).TotalMilliseconds/1000;
116:            }
117:
118:            private void saveResult (MsSql sql, string dbase, double dura-
    tion)
119:            {
120:                SqlCommand cmd = new SqlCommand();
121:
122:                cmd.CommandText = "INSERT INTO [DWH_PerformanceTest] ([Data-
    base], [Duration]) VALUES (@dbase, @duration)";
123:                cmd.Parameters.AddWithValue("@dbase", dbase);
124:                cmd.Parameters.AddWithValue("@duration", duration);
125:
126:                sql.NonQuery(cmd);
127:            }
128:        }
129:    }
130:
```

2 Datenbank Performance Test Abfragen

2.1 OLAP Abfrage

```
1: SELECT NON EMPTY {
2:     [Measures].[Plan Quantity], [Measures].[Quantity], [Measures].[Quantity
   Percent Last YTP], [Measures].[Quantity Last YTP]
3:     } ON COLUMNS, NON EMPTY {
4:           ([Zeit_Monat].[Jahr - Quartal - Monat - Da-
   tum].[Monat].ALLMEMBERS * [Zeit_Monat].[Monat Des Quartals].[Monat Des
   Quartals].ALLMEMBERS )
5:     } DIMENSION PROPERTIES MEMBER_CAPTION, MEMBER_UNIQUE_NAME ON ROWS
6: FROM [Data Warehouse DB] CELL PROPERTIES VALUE, BACK_COLOR, FORE_COLOR,
   FORMATTED_VALUE, FORMAT_STRING, FONT_NAME, FONT_SIZE, FONT_FLAGS
```

2.2 DWH Abfrage

```
1: SELECT Ist.*, LYTP.Quantity as LYTP_Quantity, LYTP.Jahr as LYTP_Jahr from
2: (
3:     (
4:           SELECT
5:           YEAR(DWH_WorkOrderLedger.Date_Key) AS Jahr, DATEPART(qq,
   DWH_WorkOrderLedger.Date_Key) AS Quartal, Month(Date_Key) as Monat,
6:           SUM(DWH_WorkOrderLedger.Quantity) AS Quantity,
   SUM(DWH_WorkOrderLedger.Plan_Quantity) as Plan_Quantity
7:           FROM DataWarehouseDB.dbo.DWH_Kostenstellen INNER JOIN
8:           DataWarehouseDB.dbo.DWH_MalfunctionCodes INNER JOIN
9:           DataWarehouseDB.dbo.DWH_WorkOrderLedger INNER JOIN
10:          DataWarehouseDB.dbo.DWH_TypeOfActivity ON
   DWH_WorkOrderLedger.[Type of Activity] = DWH_TypeOfActivity.Code ON
11:          DWH_MalfunctionCodes.Code = DWH_WorkOrderLedger.[Malfunction
   Code] ON
12:          DWH_Kostenstellen.Code = DWH_WorkOrderLedger.[Global Dimension 1
   Code]
13:          GROUP BY YEAR(DWH_WorkOrderLedger.Date_Key), DATEPART(qq,
   DWH_WorkOrderLedger.Date_Key), Month(Date_Key)
14:    ) as Ist
15:LEFT JOIN (
16:          SELECT
17:          YEAR(DWH_WorkOrderLedger.Date_Key) AS Jahr, DATEPART(qq,
   DWH_WorkOrderLedger.Date_Key) AS Quartal, Month(Date_Key) as Monat,
18:          SUM(DWH_WorkOrderLedger.Quantity) AS Quantity
```

```
19:          FROM DataWarehouseDB.dbo.DWH_Kostenstellen INNER JOIN
20:          DataWarehouseDB.dbo.DWH_MalfunctionCodes INNER JOIN
21:          DataWarehouseDB.dbo.DWH_WorkOrderLedger INNER JOIN
22:          DataWarehouseDB.dbo.DWH_TypeOfActivity ON
   DWH_WorkOrderLedger.[Type of Activity] = DWH_TypeOfActivity.Code ON
23:          DWH_MalfunctionCodes.Code = DWH_WorkOrderLedger.[Malfunction
   Code] ON
24:          DWH_Kostenstellen.Code = DWH_WorkOrderLedger.[Global Dimension 1
   Code]
25:          GROUP BY YEAR(DWH_WorkOrderLedger.Date_Key), DATEPART(qq,
   DWH_WorkOrderLedger.Date_Key), Month(Date_Key)
26:    ) as LYTP
27:    On (Ist.Jahr - 1 )= LYTP.Jahr and Ist.Quartal = LYTP.Quartal
28:)
29:order by Jahr, Quartal
```

2.3 Produktiv Datenbank Abfrage

```
1: SELECT  YEAR([Posting Month]) AS Jahr, DATEPART(qq, [Posting Month]) as
   Quartal, Month([Posting Month]) as Monat,
2:  SUM(Quantity) as Quantity, SUM(Plan_Quantity) as Plan_Quantity
3: FROM (
4:    SELECT Ist.*, Pl.Plan_Quantity FROM
5:    (
6:          select
7:                convert(datetime, CAST(YEAR([Posting Date]) AS varchar(20))
   + '-1' + '-' + CAST(MONTH([Posting Date]) AS varchar(20))) AS [Posting
   Month],
8:                [Type of Activity], [Global Dimension 1 Code], [Malfunction
   Code],
9:                SUM(Quantity) as Quantity
10:
11:          from dbo.[CRONUS TEST SE DD GmbH$Work Order Ledger Entry] as WOLE
12:          LEFT JOIN dbo.[CRONUS TEST SE DD GmbH$Work Order Header] as WOH
   ON WOH.[No_] = WOLE.[Work Order No_]
13:          GROUP BY YEAR([Posting Date]), DATEPART(qq, [Posting Date]),
   Month([Posting Date]),
14:          [Type of Activity], [Global Dimension 1 Code], [Malfunction Code]
15:    )
16:    AS Ist
17:    LEFT JOIN
18:    (
```

```
19:          SELECT (Quantity) as Plan_Quantity, [Malfunction Code], [Type of
   Activity], [Date_Key]
20:          from
21:          (
22:                SELECT Quantity, Pw.[Malfunction Code], Pw.[Type of Activi-
   ty]
23:                  , [Date_Key], Anzahl.Anz as Anzahl
24:                FROM [DataWarehouseDB].[dbo].[Planwert] as Pw
25:                left join
26:                (
27:                      select
28:                            convert(datetime, CAST(YEAR([Posting Date])
   AS varchar(20)) + '-1' + '-' + CAST(MONTH([Posting Date]) AS varchar(20)))
   AS [Posting Month],
29:                            COUNT(*) as Anz, [Type of Activity], [Mal-
   function Code]
30:                      from dbo.[CRONUS TEST SE DD GmbH$Work Order Ledger
   Entry] as WOLE
31:                      LEFT JOIN dbo.[CRONUS TEST SE DD GmbH$Work Order
   Header] as WOH ON WOH.[No_] = WOLE.[Work Order No_]
32:                      GROUP BY YEAR([Posting Date]), DATEPART(qq, [Posting
   Date]), Month([Posting Date]),
33:                      [Type of Activity], [Malfunction Code]
34:                )
35:                as Anzahl
36:                ON (
37:                      Anzahl.[Type of Activity] = Pw.[Type of Activity]
   collate Latin1_General_CS_AS
38:                      and Anzahl.[Malfunction Code] = Pw.[Malfunction
   Code] collate Latin1_General_CS_AS
39:                      and Anzahl.[Posting Month] = Pw.[Date_Key]
40:                )
41:          ) Plan_Calc
42:    )
43:    As Pl
44:    ON
45:    Ist.[Malfunction Code] = Pl.[Malfunction Code]  collate Lat-
   in1_General_CS_AS
46:          and Ist.[Type of Activity] = Pl.[Type of Activity] collate Lat-
   in1_General_CS_AS
47:          and Ist.[Posting Month] = Pl.[Date_Key]
48:) Res
49:GROUP BY YEAR([Posting Month]), DATEPART(qq, [Posting Month]),
   Month([Posting Month])
```

```
50: order by YEAR([Posting Month]), DATEPART(qq, [Posting Month]),
    Month([Posting Month])
```

Eigenständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt.

Ort, Datum Vorname Nachname